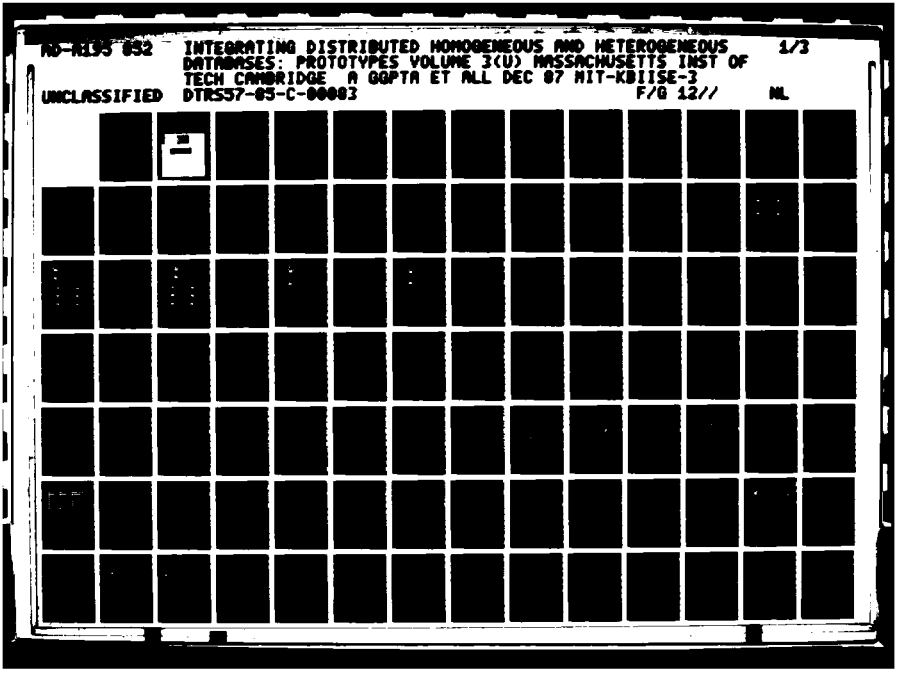
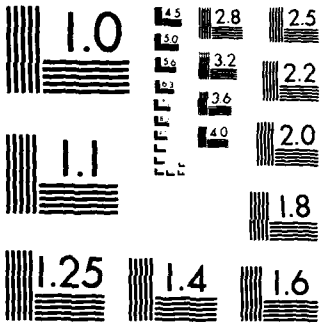

AD-A195 852
INTEGRATING DISTRIBUTED HOMOGENEOUS AND HETEROGENEOUS DATABASES: PROTOTYPES VOLUME 3(U) MASSACHUSETTS INST OF TECH CAMBRIDGE A GUPTA ET AL DEC 87 MIT-KBIISE-3
1/3
UNCLASSIFIED
DTRS57-85-C-00083
F/G 12//
NL



Massachusetts
Institute of
Technology

Knowledge-Based
Integrated Information
Systems Engineering
(KBIISE) Project

Volume 3

Integrating Distributed Homogeneous and Heterogeneous Databases: Prototypes

(2)

Amar Gupta
Stuart Madnick
SERIES EDITORS

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: KBIISE-3	2. GOVT ACCESSION NO. AD-A188 552
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: Integrating Distributed Homogeneous and Heterogeneous Databases: Prototypes	5. TYPE OF REPORT & PERIOD COVERED: Part of Final Report; Sept. 86 - Jan. 88
6. PERFORMING ORG. REPORT NUMBER: KBIISE-3	
7. AUTHOR(s): Amar Gupta and Stuart Madnick (Editors)	8. CONTRACT OR GRANT NUMBER(s): DTRS57-85-C-00083
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Massachusetts Institute of Technology, Cambridge, MA 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Transportation Systems Center, Broadway, MA 02142 (In conjunction with U.S. Air Force)	12. REPORT DATE: December 1987
	13. NUMBER OF PAGES: 218 Pages
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: Transportation Systems Center, Broadway, MA 02142	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

This volume is one of eight volumes prepared by M.I.T. for Department of Transportation and Department of Defense (U.S. Air Force).

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Integration, knowledge, databases, systems engineering, methodologies, information modeling, heterogeneous database systems

20 ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This volume discusses key issues relating to distributed databases, and presents alternate methods for integrating them together. It is divided into four parts. The first part, "evolution Towards Strategic Applications of Databases Through Composite Information Systems," divides applications into four categories: inter-corporate, inter-divisional, inter-product, and inter-model. The process of evolution is described in terms of five stages: separate systems, virtual-terminal driver, logical separation, physical

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

separation, and specialized functional engine.

The second part, "Distributed Homogeneous Database systems: A Comparison between Oracle and Ingres," compares commercial products in terms of the levels of transparency and independence supported by them. Six properties of transparency and five properties of independence are identified. In spite of significant research activities, neither of these products are able to meet all the requirements.

The third part, "Achieving a Single Model for Integrating Heterogeneous Databases attempts to come up with a single unified model that encompasses both the database issue and the communication issue. In the communication area, there are types of standards: connection-oriented and connectionless. In the database area there are multiple standards suited for different environments.A single model that can consolidate these alternatives would produce a more manageable situation.

The fourth part, "A Technical Comparison of Distributed Heterogeneous Database Management Systems, " describes eight systems being developed around the world. Because of the added complexity involved in translating between multiple systems and multiple data models, distributed heterogeneous database systems are more complex than equivalent homogeneous ones. While all of these eight systems are able to do global retrieves, their ability to perform global updates and other capabilities is varied.

INTEGRATING DISTRIBUTED HOMOGENEOUS AND HETEROGENEOUS DATABASES - PROTOTYPES

Amar Gupta
Stuart Madnick

Series Editors

Knowledge-Based Integrated Information Systems Engineering (KBIISE) Report: Volume 3

Massachusetts Institute of Technology

INTEGRATING DISTRIBUTED HOMOGENEOUS AND HETEROGENEOUS DATABASES - PROTOTYPES

About This Volume

This volume discusses key issues relating to distributed databases, and presents alternate methods for integrating them together. It is divided into four parts. The first part, "Evolution Towards Strategic Applications of Databases Through Composite Information Systems," divides applications into four categories: inter-corporate, inter-divisional, inter-product, and inter-model. The process of evolution is described in terms of five stages: separate systems, virtual-terminal driver, logical separation, physical separation, and specialized functional engine.

The second part, "Distributed Homogeneous Database systems: A Comparison between Oracle and Ingres," compares commercial products in terms of the levels of transparency and independence supported by them. Six properties of transparency and five properties of independence are identified. In spite of significant research activities, neither of these products are able to meet all the requirements.

The third part, "Achieving a Single Model for Integrating Heterogeneous Databases" attempts to come up with a single unified model that encompasses both the database issue and the communication issue. In the communication area, there are two types of standards: connection-oriented and connectionless. In the database area there are multiple standards suited for different environments. A single model that can consolidate these alternatives would produce a more manageable situation.

The fourth part, "A Technical Comparison of Distributed Heterogeneous Database Management Systems," describes eight systems being developed around the world. Because of the added complexity involved in translating between multiple systems and multiple data models, distributed heterogeneous database systems are more complex than equivalent homogeneous ones. While all of these eight systems are able to do global retrieves, their ability to perform global updates and other capabilities is varied.

Keywords: Knowledge Based Integrated Information Systems Engineering (KBIISE), I(KR)

Table of Contents

	Page
SERIES EDITORS' NOTE	1
EVOLUTION TOWARDS STRATEGIC APPLICATIONS THROUGH COMPOSITE INFORMATION SYSTEMS (Technical Report #4)	5
DISTRIBUTED HOMOGENEOUS DATABASE SYSTEMS: A COMPARISON BETWEEN ORACLE AND INGRES (Technical Report #11)	31
ACHIEVING A SINGLE MODEL FOR INTEGRATING HETEROGENEOUS DATABASES (Technical Report #16)	127
A TECHNICAL COMPARISON OF DISTRIBUTED HETEROGENEOUS DATABASE MANAGEMENT SYSTEMS (Technical Report #5)	159

Knowledge-Based Integrated Information System Engineering Project: Volume 3
Amar Gupta and Stuart E. Madnick, Editors

DEDICATED

TO

THE

NEXT

GENERATION

OF

PROFESSIONALS

SERIES EDITORS' NOTE

This book is one of eight volumes published by MIT as part of the Knowledge-Based Integrated Information Systems Engineering Project (KBIISE). In order to appreciate the papers in this book, it is necessary to be aware about the theme of the KBIISE project, its major objectives, and the different documents that summarize the research accomplishments to date.

Goal

The primary goal of the KBIISE project is to integrate islands of disparate information systems that characterize virtually all large organizations. The number and the size of these islands has grown over years and decades as organizations have invested in an increasing number of computer systems to support their growing reliance on computerized data. This has made the problem of integration more pronounced, complex, and challenging.

The need for multiple systems in large organizations is dictated by a combination of technical reasons (such as the desired level of processing power and the amount of storage space), organizational reasons (such as each department obtaining its own computer based on its function), and strategic reasons (such as the level of reliability, connectivity, and backup capabilities). Further, underlying trends in the information technology area have led to a situation where most organizations now depend on a portfolio of information processing machines, ranging from mainframes to minicomputers and from general purpose workstations to sophisticated CAD/CAM systems, to support their computational requirements. The tremendous diversity and the large size of the different systems make it difficult to integrate these systems.

Key Participants

The above problem is becoming increasingly evident in all large government agencies and in large development programs. In the fall of 1986, the U.S. Air Force (USAF) and the Transportation Systems Center (TSC) of the U.S. Department of Transportation approached M.I.T. to conduct and to coordinate research activity in this area in order "to develop the framework for a comprehensive methodology for large scale distributed, heterogeneous information systems which will provide: (i) the necessary structure and standards for an evolving top down global framework; (ii) simultaneous bottom up systems development; and (iii) migratory paths for existing systems."

Both USAF and TSC provided sustained assistance to members of our research team. In addition, Citibank and IBM provided some funds for research in very specific areas. One advantage of our corporate links was the opportunity to analyze and to generate case studies of actual decentralized organizational environments.

The research sponsors and MIT agreed that in order to deal with the heterogenity issue in a meaningful way, it was important that a critical mass of influential individuals participate in the development of solutions. Only through widespread discussion and acceptance of a proposed strategy would it become feasible to deal with the major problems. For these reasons, a Technical Advisory Panel (TAP) was constituted. Nominees to the TAP included experts from academic and research organizations, government agencies, computer companies, and other corporations. In addition, several subcontractors, the primary one being Texas A&M University, provided assistance in specific areas.

Technical Outputs

The scope of the work included (i) technical issues; (ii) organizational issues; and (iii) strategic issues. On the basis of exploratory research efforts in all these areas, 24 technical reports were prepared. Eighteen of these reports were generated by MIT research personnel, and their respective areas of investigation are summarized in the figure on the opposite page.

The five technical reports, not represented in the figure, are as follows:

#1. Summary.

#2. Record of discussions held at the first meeting of the Technical Advisory Panel (TAP) on February 17, 1987.

#3. Consolidated report submitted by Texas A&M University.

#21. Annotated Bibliography.

#23. Record of discussions held at the second meeting of the Technical Advisory Panel (TAP) on May 21 and 22, 1987.

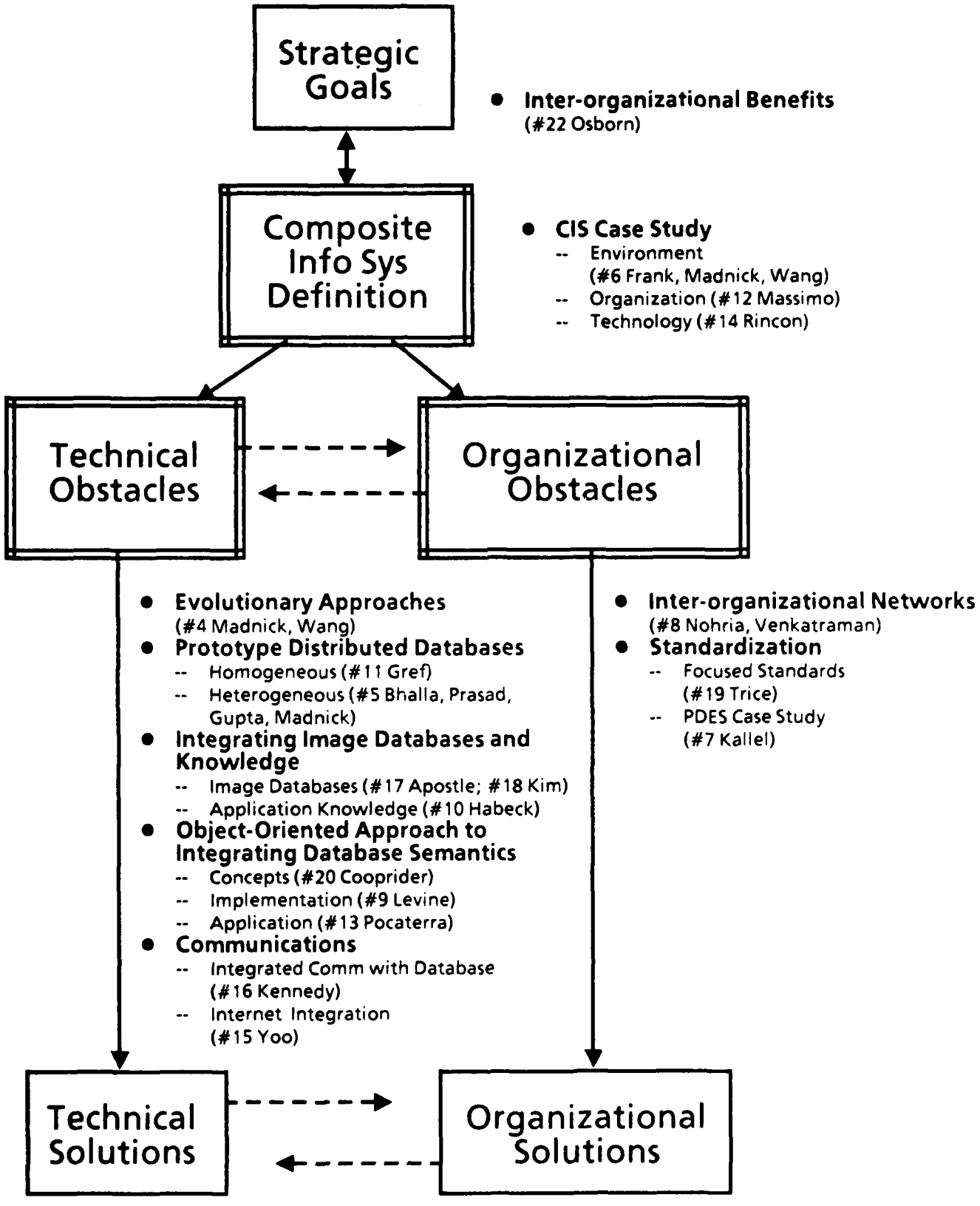
#24 Contributions received from members of the TAP highlighting their views on various aspects of the problem.

All the 24 technical reports have been edited and reorganized as an eight-volume set. The titles of the different volumes are as under:

1. KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS ENGINEERING-HIGHLIGHTS AND BIBLIOGRAPHY
2. KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS DEVELOPMENT METHODOLOGIES PLAN
3. INTEGRATING DISTRIBUTED HOMOGENEOUS AND HETEROGENEOUS DATABASES - PROTOTYPES
4. OBJECT-ORIENTED APPROACH TO INTEGRATING DATABASE SEMANTICS
5. INTEGRATING IMAGES, APPLICATIONS, AND COMMUNICATIONS NETWORKS
6. STRATEGIC, ORGANIZATIONAL, AND STANDARDIZATION ASPECTS OF INTEGRATED INFORMATION SYSTEMS
7. INTEGRATING INFORMATION SYSTEMS IN A MAJOR DECENTRALIZED INTERNATIONAL ORGANIZATION
8. TECHNICAL OPINIONS REGARDING KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS ENGINEERING

Volume 2 contains the report submitted by Texas A&M and Volume 8 highlights the views of members of the TAP. Activities described in the other 6 volumes have been conducted at MIT.

EXPLORATORY RESEARCH EFFORTS



Acknowledgments

Funds for this project have been provided by U.S. Air Force, U.S. Department of Transportation (Contract Number DTRS57-85-C-00083), IBM, and Citibank. We thank all these organizations and their representatives for their support. In particular, we are indebted to Major Paul Condit of U.S. Air Force for his initiative in sponsoring this project, to Dr. Frank Hassler, Bud Giangrande, and Bob Berk of the Transportation Systems Center (TSC) for their support and assistance, to Professor Joseph Sussman, Director, Center for Transportation Studies (CTS) at MIT for his help and encouragement, and to all the individuals whose results have been published in this book.

We would welcome receiving feedback from readers of this book.

Amar Gupta and S.E. Madnick
Massachusetts Institute of Technology
Cambridge, Massachusetts.

EVOLUTION TOWARDS STRATEGIC APPLICATIONS THROUGH COMPOSITE INFORMATION SYSTEMS

STUART MADNICK AND Y. RICHARD WANG

One important category of strategic applications involves inter-organizational linkage (e.g., tying supplier and/or buyer systems) or intra-organizational integration (e.g., tying together disparate functional areas of an organization). Applications in this category require multiple very large databases to work together to support the business activities. Information systems in this category are referred to in this paper as Composite Information Systems (CIS). Four categories of potential CIS applications have been identified:

1. Inter-organizational - which involve two or more separate organizations (e.g., direct connection between production planning system in one company and order entry system in another company).
2. Inter-divisional - which involves two or more divisions within a firm (e.g., corporate-wide coordinated purchasing).
3. Inter-product - which involves the development of sophisticated information services by combining simpler services (e.g., a cash management account that combines brokerage services, checks, credit card, and savings account features).
4. Inter-model - which involves combining separate models to make more comprehensive models (e.g., combine economic forecasting model with optimal distribution model to analyze the impact of economic conditions on distribution).

The challenge is that it requires inter-disciplinary expertise (e.g., database management, data communication, systems engineering, organizational development, and strategic management) to develop and/or to deploy information resources within and/or across organizational boundaries to facilitate corporate strategic goals.

An approach is proposed in this paper as an interim step to meet the challenge. The essence of this approach is captured in four CIS principles: 1) the explicit recognition of the CIS environment; 2) the separation of data from processing; 3) the use of flexible tools; and 4) the use of interfaces that facilitate data conversion and communication between processing components and databases.

Migrating from a non-integrated environment to an integrated environment is usually a difficult, expensive, and time-consuming process both due to technical dufficulties as well as organizastional realities. Thus, an evolutionary approach is desirable, if not critical.In this research effort five stages of evolution, which may co-exist, have been identified as follows:

1. Separate systems. This is the assumed starting point. The only communication among the separate systems is through human users.
2. Virtual terminal driver. Existing terminal protocols are driven by a *CIS executive*. There is no need to modify the systems in order to interface with the CIS executive.
3. Logical separation. As new applications are developed the data and processing are logically separated.
4. Physical separation. At this stage "file servers" and "data base servers" are used to physically separate the processing from the data -- further encouraging sharing of the databases.
5. Specialized functional engine. As the technology becomes available, specialized high-performance and high-availability data base machines can be used to replace the "data base servers" and to serve a large and diverse community -- producing, in essense, an *information utility*.

The opportunities for strategic uses of information technology in organizations is often blocked by the difficulties of getting from "what is" to "what is desired." The five-stage evolutionary process presented in this paper has been found to be effective in overcoming this problem.

TECHNICAL REPORT #4

1. Introduction

Significant advances in the price, speed-performance, capacity, and capabilities of new information technologies have created a wide range of opportunities for business applications. These opportunities can be exploited to meet corporate information needs and to gain strategic advantage. In this paper, the concept of *Composite Information Systems* (CIS) is presented as an approach in the evolution towards databases which can be deployed by companies to implement competitive strategies. This approach provides a framework for the evolution of separate systems to a more fully integrated system with value being added to the organization at each stage.

2. Strategic CIS Opportunities Using Databases

Development and deployment of information systems for strategic computing have become very topical [4, 5, 7, 13, 19, 21, 22]. Porter [20] found that information technology is changing the rules of competition for U.S. industry by: 1) changing industry structure and boundaries; 2) dramatically reducing costs, thereby, creating competitive advantage; and 3) creating new products and services, sometimes spawning completely new business.

In the database arena, much research has been conducted on the design of large capacity, cost-effective memory systems with rapid access time [10, 11, 17, 18, 23]. In the private sector, commercial database machines, such as Britton Lee's IDM 500 and Teradata's DBC 1012 [6], have been introduced. These ideas and products can be very important for implementing systems to facilitate corporate strategic goals.

One important category of strategic database applications involves inter-corporate cooperation (e.g., tying into supplier's and/or buyer's systems) or intra-corporate integration (e.g., tying together disparate functional areas of a firm). Applications in this category require a collection of databases to work together to support the business activities. The authors refer to information systems in this category as *Composite Information Systems* (CIS).

CIS applications require the deployment and/or development of sophisticated communication networks to support connectivity among diverse applications. The challenge is that this requires inter-disciplinary expertise (e.g., database management, data communication, systems engineering, organizational development, and strategic management) to develop and/or to deploy information resources within and/or across organizational boundaries to facilitate corporate strategic goals.

Four categories of potential CIS applications have been identified: 1) inter-corporate, 2) inter-divisional, 3) inter-product, and 4) inter-model. The following subsections exemplify CIS applications that facilitate strategic goals.

2.1 Inter-Corporate Applications

American Hospital Supply (AHS), a manufacturer and distributor of a broad line of products for doctors, laboratories, and hospitals, has since 1976 evolved an order entry/distribution system that directly links the majority of its customers to AHS computers. Over 4,000 customer sites are linked to the AHS system (i.e., an inter-corporate application). As well as providing the customer with direct access to the AHS order/distribution process, the system supports many customer functions, such as inventory control. The AHS system has been successful because it simplifies the ordering process for customers, reduces costs for both AHS and the

customer, and allows AHS to develop and manage pricing incentives to the customer across all product lines. As a result, customer loyalty is high and AHS, which started out as a fairly small company, has gained a significant market share [2].

2.2 Inter-divisional Applications

A major automobile manufacturer discovered, in the late 70's, that the average cost of certain components used widely by many divisions of the company could be reduced by two thirds if it could buy these components in bulk from a much smaller set of suppliers. Cost savings would result if timely information concerning each division's production plans and inventory levels could be obtained so that a corporate-wide plan of purchase could be implemented (i.e., an inter-divisional application). However, these divisional information systems were all designed and maintained separately. A CIS capable of accessing all the pertinent information across divisions would provide timely information to reduce ordering and inventory costs.

2.3 Inter-Product Applications

In 1977, Merrill Lynch established the Cash Management Account (CMA) which shattered the traditional boundaries between the banking and securities industries. The CMA account is an integration of brokerage service, VISA debit card, and checking account (i.e., inter-product application). Implementation required a complex interface of telecommunication and database management systems. With the CMA account, Merrill brought in over 450,000 new accounts, reaped over $60 Million a year in fees, and dominated the market for four years.

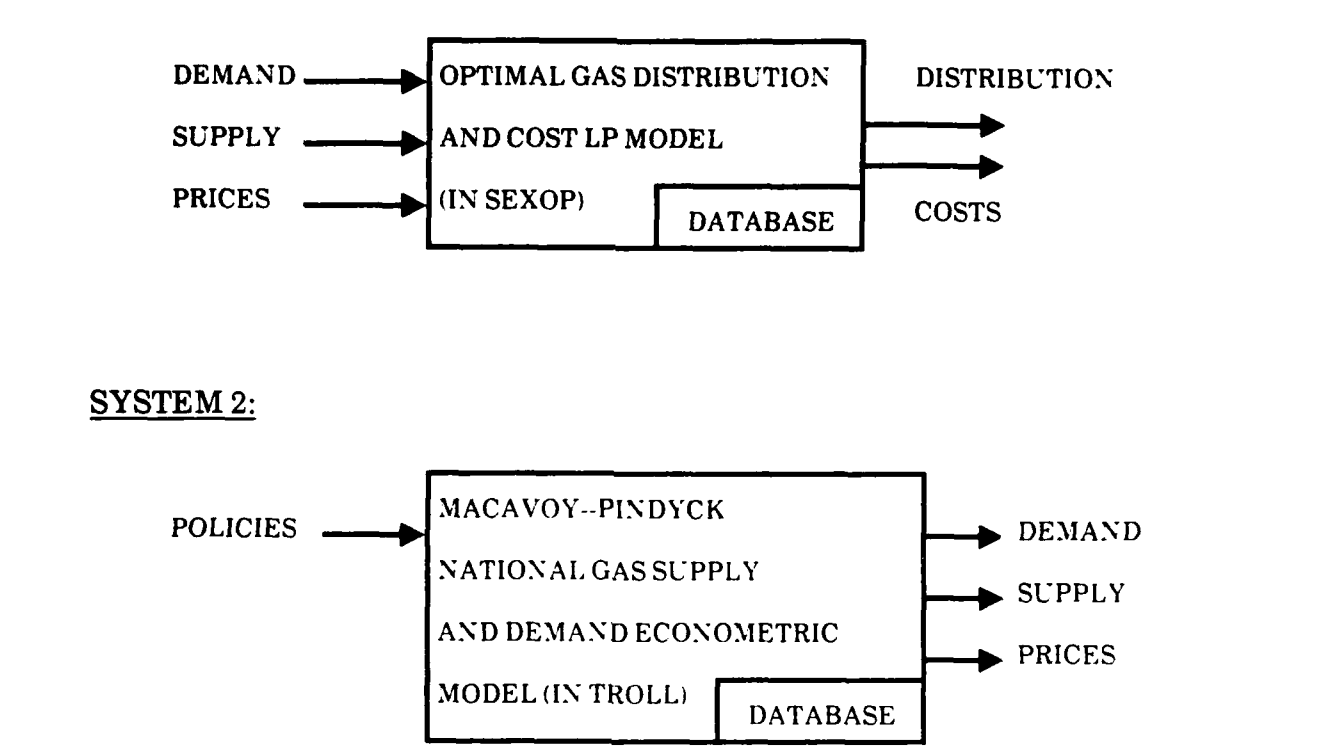
Competition from other financial services organizations did not begin to appear until 1981 [24].

2.4 Inter-Model Applications

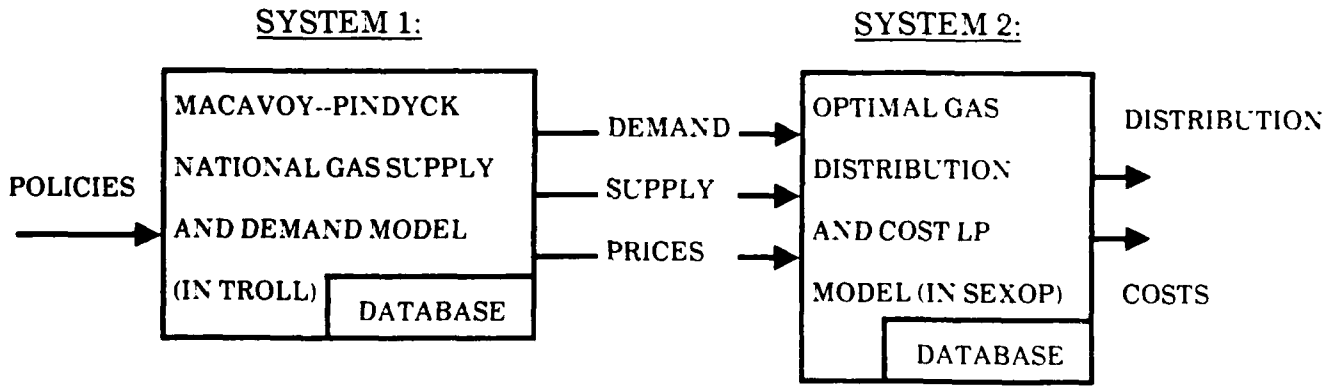
As part of energy policy analysis research at MIT, the MacAvoy-Pindyck gas model, using the TROLL econometric modeling system, was developed to study the impact of government policies on the demand, supply, and prices of natural gas. To determine the optimal distribution schedule and its cost, the New England Regional Commission was interested in combining its Decision Support System (DSS), using the SEXOP linear programming optimization system, with the MacAvoy-Pindyck system, as shown in Figure 1, to explore the impact of the various government policies on profits. This was a major challenge since the two systems had been developed independently with different tools, languages, and databases.

2.5 Executives' Obstacle

Madnick and Wang [19] have interviewed hundreds of executives making IS decisions and found that the majority of the executives are very concerned about the need to deploy CIS to gain strategic advantage but have difficulty in approaching the problem, recognizing the nature of the problem, and linking issues such as technical incompatibility, organizational standardization, and strategic allocation. In these studies, need for connectivity (a CIS subgoal) was noted to be a major concern facing these executives. On a ten-point scale of importance, it was rated 8.2 and over 40% of the executives noted that it was their most important problem.



(A) Independent Modeling Systems



(B) Desired Composite Modeling System

Figure 1. An Energy DSS Example

"As top managers have been attracted to information as a powerful source, it is only natural that information systems should get bigger," suggested Appleton [1] in describing a project with estimates of 170,000 man-hours, $10 million, and five elapsed years. "The project manager was bewildered in making the transition from the old separate systems to the new desired system - how long it would take, how much of the old systems he should leave in place, how does he continue to provide value to his organization on an annual basis, etc.," said Appleton. The CIS approach presented below provides a framework for the evolution of separate systems to a more fully integrated system with value being added to the organization at each stage.

3. Principles of CIS

As pointed out earlier, a composite information system is a system which integrates "independent" systems which may reside within and/or across organizational boundaries. By "independent" we mean systems which are (or were) developed independently, usually by separate groups or organizations. It is crucial to realize that the "independence" of these systems is not necessarily a mistake. It is often driven by needs for division of responsibility, organizational autonomy, and/or differences in objectives. However, it may be important to access these systems in concert for certain purposes.

The traditional approach to system development, not sensitive to the synergistic issue, tends to result in sealed systems as depicted in Figure 2. The process, model, or tool of system 1 do not communicate with those of system 2. Cross-access of algorithms and data under this approach is practically impossible. The CIS approach facilitates cooperation between systems by following certain principles in the system design process. The essence of this approach is captured in the

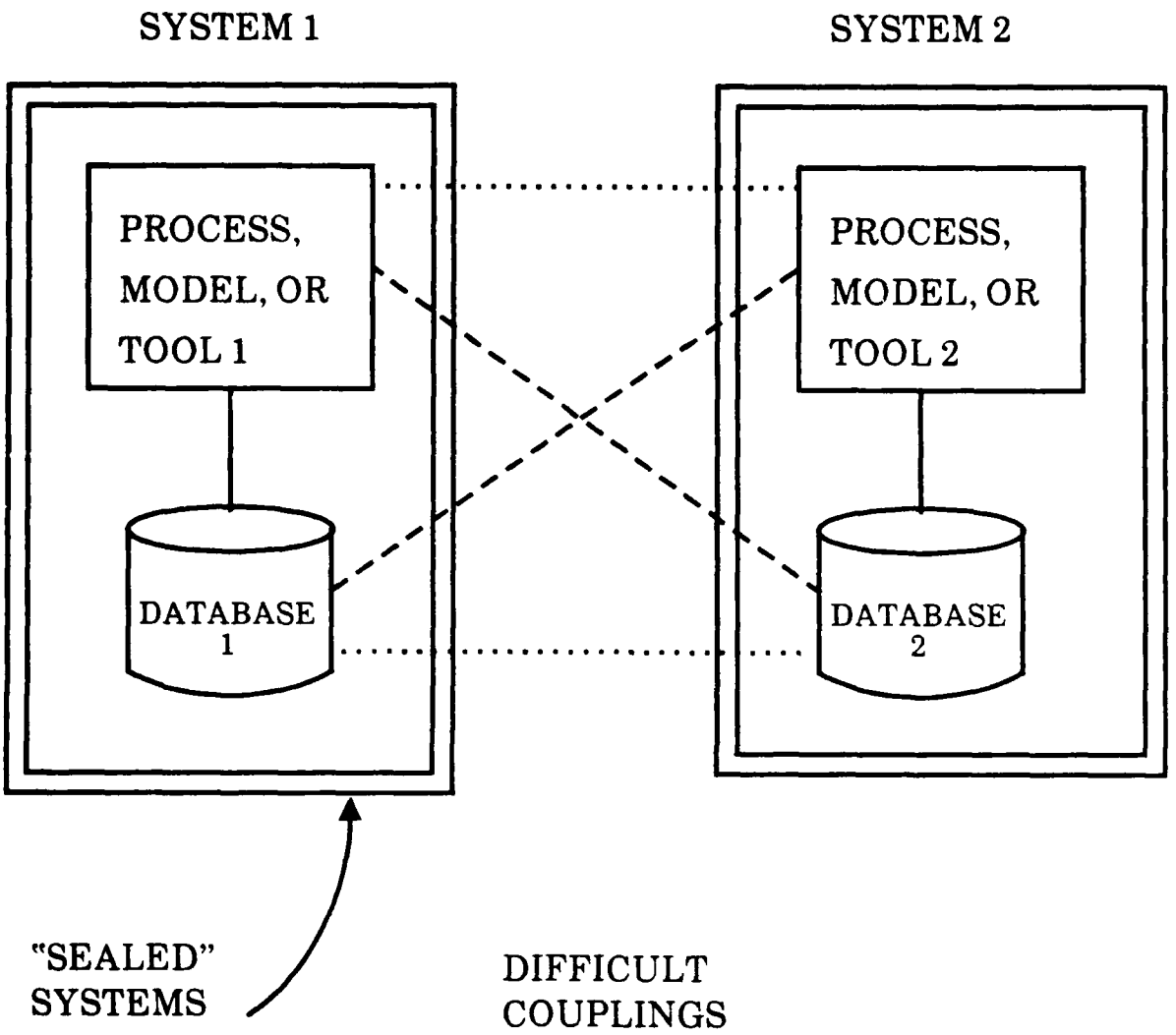


Figure 2. The traditional approach to system development.

explicit recognition of the following four CIS principles: 1) the CIS environment principle; 2) the data separation principle; 3) the tool development principle; and 4) the interface composition principle.

3.1 The CIS Environment Principle

This principle addresses the need to explicitly recognize that it is important to allow for the coexistence and usage of a variety of components (e.g., different types of database systems, models, and applications). Actively and explicitly supporting such multiplicity is a key goal of the CIS approach.

3.2 The Data Separation Principle

This principle provides a logical separation of the database from processing. The separation enables two "windows" to be opened up, i.e., the process descriptor and the database descriptor. The process descriptor describes the name, the input/output data requirement, and other resource requirements of the processing components. The database descriptor contains information about the data (e.g., data model, schema, access rights) in the database. These two descriptors can be used by the execution environment to coordinate the interaction between the processing component and the database.

Flexibility should also be carefully designed into the database so the information in the database can be viewed from different angles to serve multiple purposes. This allows the database to be accessed by other systems implemented independently. Moreover, new types of information or relationships can be added to the database easily as the database evolves.

3.3 The Tool Development Principle

This principle advocates the usage of a set of software tools, such as special purpose languages, to facilitate the construction of applications. For instance, TROLL is an econometrics model construction language, and TSP is a time series analysis model construction language. These languages provide more specialized, higher-level primitives than traditional general purpose languages (i.e, they are general purpose tools). By allowing applications to be constructed from the same general purpose tools, inter-application communication protocols, which may be cumbersome to implement in the general purpose languages, can be streamlined.

3.4 The Interface Composition Principle

This principle allows interface mechanisms to be built for data conversion and communication between processing components and databases. Three types of interface mechanisms, as shown in Figure 3, have been identified: BRIDGE, LINK, and SPAN.

BRIDGE provides the necessary conversion of arguments to allow invocation of a processing component from another. LINK provides a mapping between two databases with dissimilar types of data models. SPAN converts the data format retrieved from the database into the processing format (and vice versa).

The CIS Executive, as shown in Figure 3, coordinates the user, BRIDGE, LINK, SPAN, the processing component, and the database. It directs the request of data from a processing component to the target database, and invokes appropriate interface routines, if necessary, to convert the data, and returns the result to the

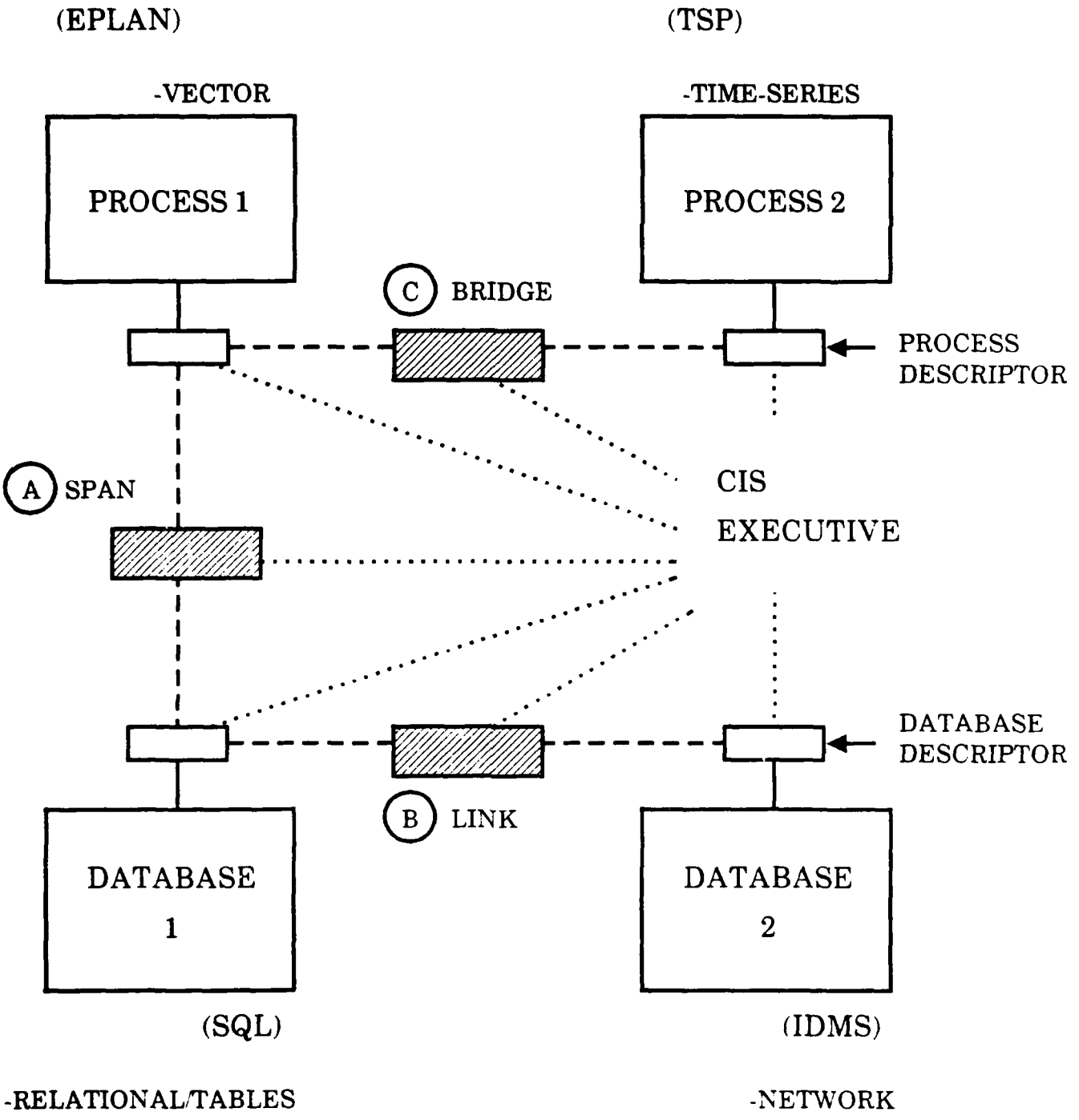


Figure 3. The CIS approach to system development

processing component. It also invokes a processing component on behalf of another.

We have described the four CIS principles for strategic applications. These principles have been applied to integrate separate systems using concepts such as virtual machines, common systems, and special execution environments [14].

4 Evolving CIS: An Implementation Strategy

In the study of IS executives reported by Madnick and Wang [19], a three phase evolutionary process of inter and intra organizational computing was identified, as shown in Figure 4. Networking (phase 1) provides the necessary backbone structure for the connectivity (phase 2) of diverse, often incompatible, systems which in turn makes available opportunities for strategic computing (phase 3).

This paper focuses its attention on how an organization may engage in a staged development of CIS to evolve through phase 2 (connectivity). Five stages of connectivity have been identified as follows: 1) separate systems, 2) the virtual-terminal driver, 3) the logical separation of processing from database, 4) the physical separation of processing from database, and 5) the specialized functional engines.

4.1 Separate Systems (Stage 1)

The initial stage consists of a set of existing systems that either do not communicate with each other or, more typically, only communicate via human operators, as shown in Figure 5. The processing component and the database component of each system are tightly coupled. The only existing "window" is the user interface via the teminal.

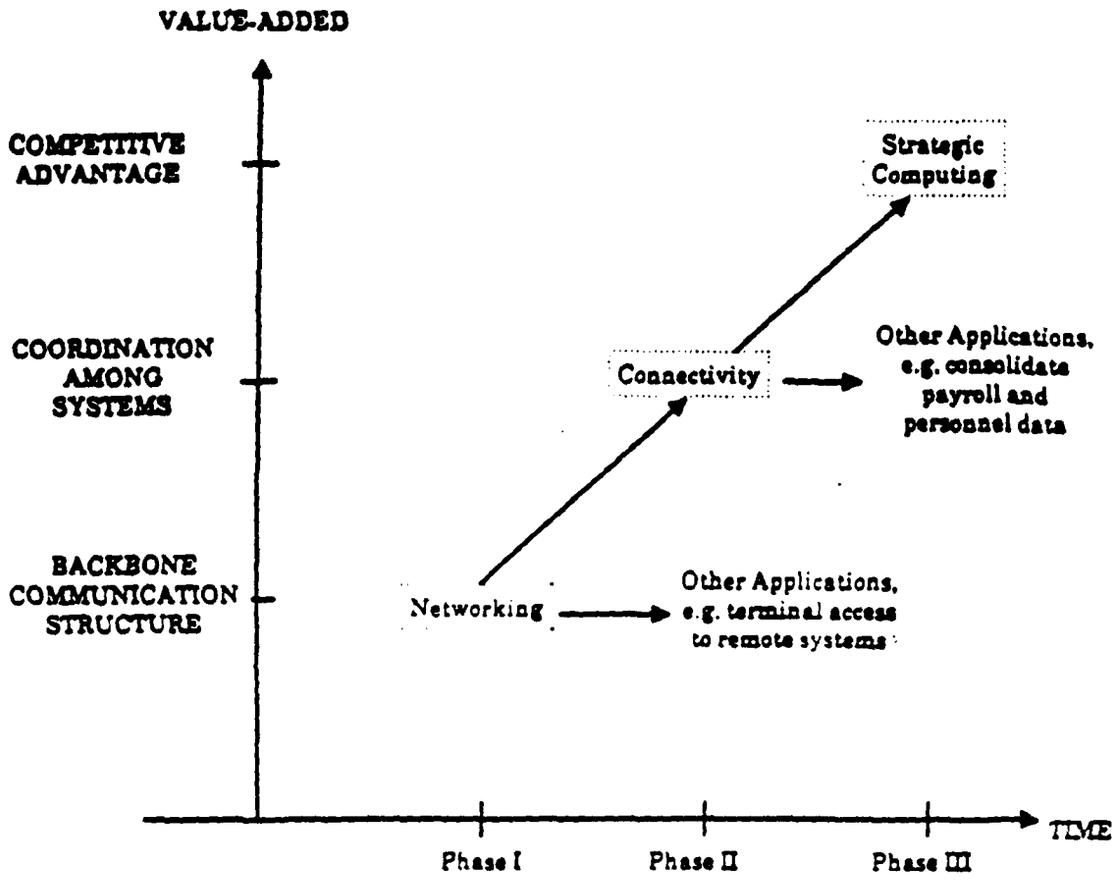


Figure 4. Evolution of Inter and Intra Organizational Computing

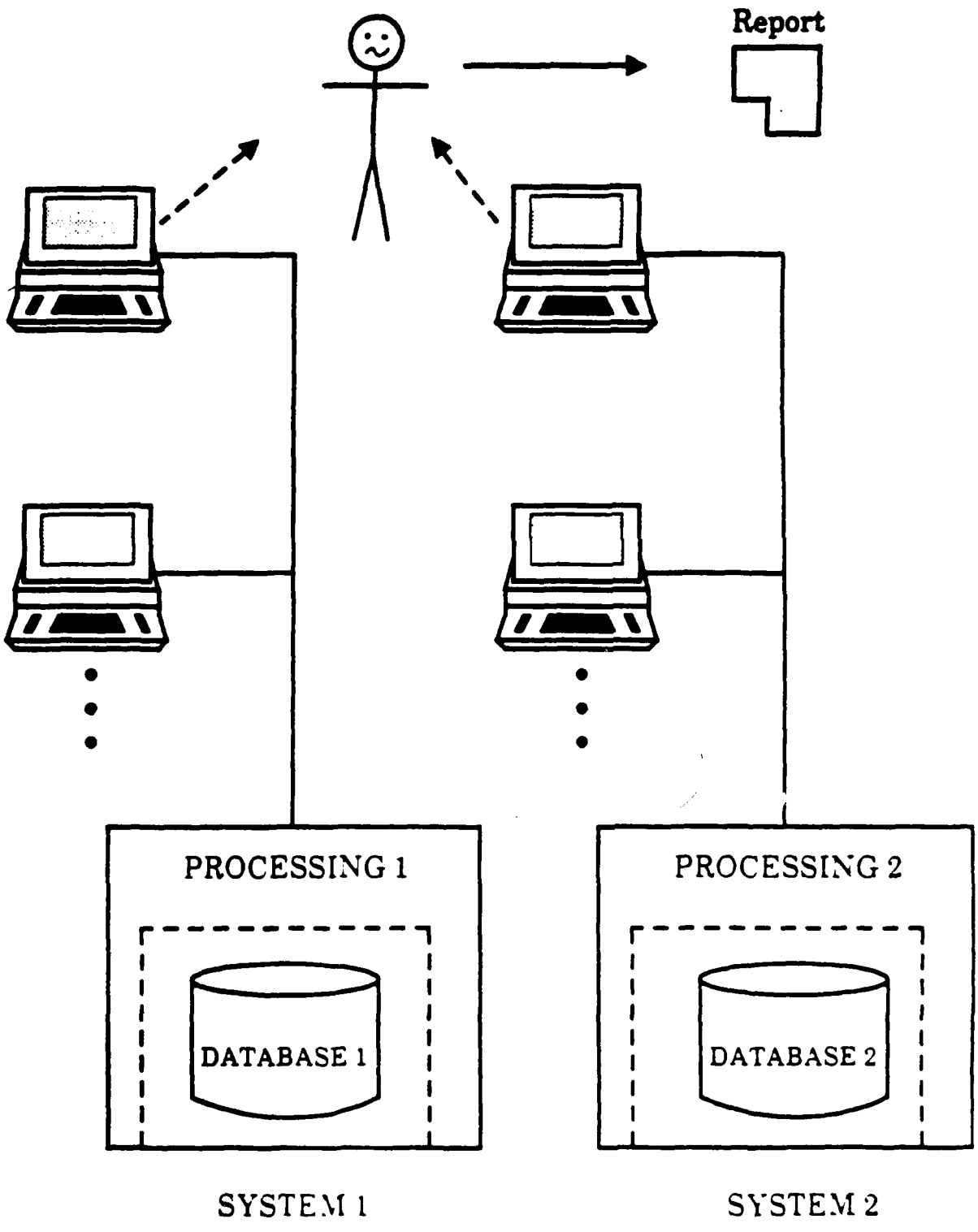


Figure 5. Separate systems (Stage 1)

4.2 Virtual-Terminal Driver (Stage 2)

The existing terminal protocols are used to interface existing systems, as shown in Figure 6. *Virtual-terminals* are created which are indistinguishable to the system from real terminals. The real terminals can still be used to perform their traditional functions. A customer interested in his other composite account status may invoke the CIS Executive which "drives" the virtual-terminals. The Executive invokes each system (via its virtual-terminal) to obtain the necessary information. Incompatibilities between the account data in the two systems are resolved by the Executive which then presents a composite answer to the customer.

As an example, a UNIX based professional workstation (or personal computer) has been used in several recent applications to link separate systems. Using UNIX as base for the CIS Executive and its CU command to simulate the virtual terminal, it is possible to dial into multiple remote disparate systems. The UNIX workstation appears as a virtual terminal to each of the remote systems. The customer interested in his composite account status invokes a SHELL script which sends the appropriate terminal sequences to each system (via CU), receives the resulting responses (via UNIX "pipes") resolves any incompatibilities between the account data, and finally presents the composite answer to the customer.

The virtual-terminal concept is very powerful in connecting separate systems. Very few changes, if any, need to be made to the existing systems, and construction of the Executive is relatively straight forward. Therefore, a CIS using the Executive approach can be brought up in a relatively short period of time.

As a recent example, four banks in the mid-Atlantic states merged, each had developed its own very different account status systems (e.g., Burrough, IBM, etc.).

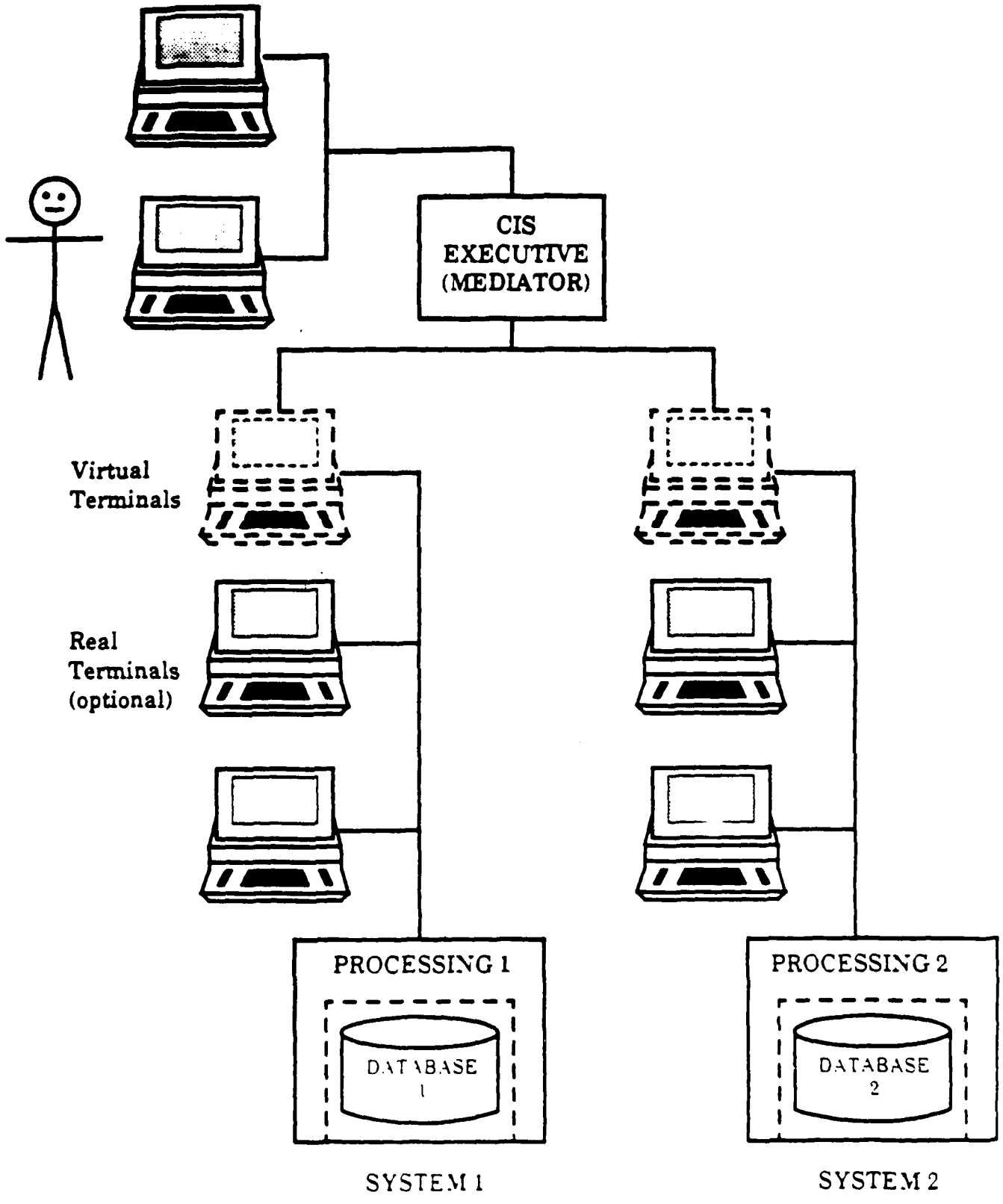


Figure 6. CIS Executive (Stage 2)

To maximize their new market power, it was critical to provide a single coherent account status system rapidly. Using the virtual terminal concept, this was accomplished within a month. This capability is quite important because it can provide functional benefit to the organization quickly and, thereby, sustain upper management support to continue the evolution.

The major drawback of the virtual terminal driver approach is that it remains difficult to access the databases, which are sealed in each system, for purposes not supported through terminal commands. Adding new functions and new types of data is very cumbersome. This leads to the rationale for logical separation (stage 3).

4.3 Logical Separation (Stage 3)

As the organization evolves, one or more of the systems will need to be *significantly revised (and/or new* systems developed) to meet changing business needs and to keep the systems operationally efficient. At such a point, the CIS principles described in section 3 can be applied. In particular, logical separation of the processing component from its database should be designed into the systems, as shown in Figure 7. By installing a database management package, the database activities are offloaded from the processing component of the system. This database is also made available to the CIS Executive through the database interface. A dotted line connecting the CIS Executive to the database (see Figure 7) represents new uses of the database by the CIS Executive. Multiple subsystems may go through this transformation as the system evolves.

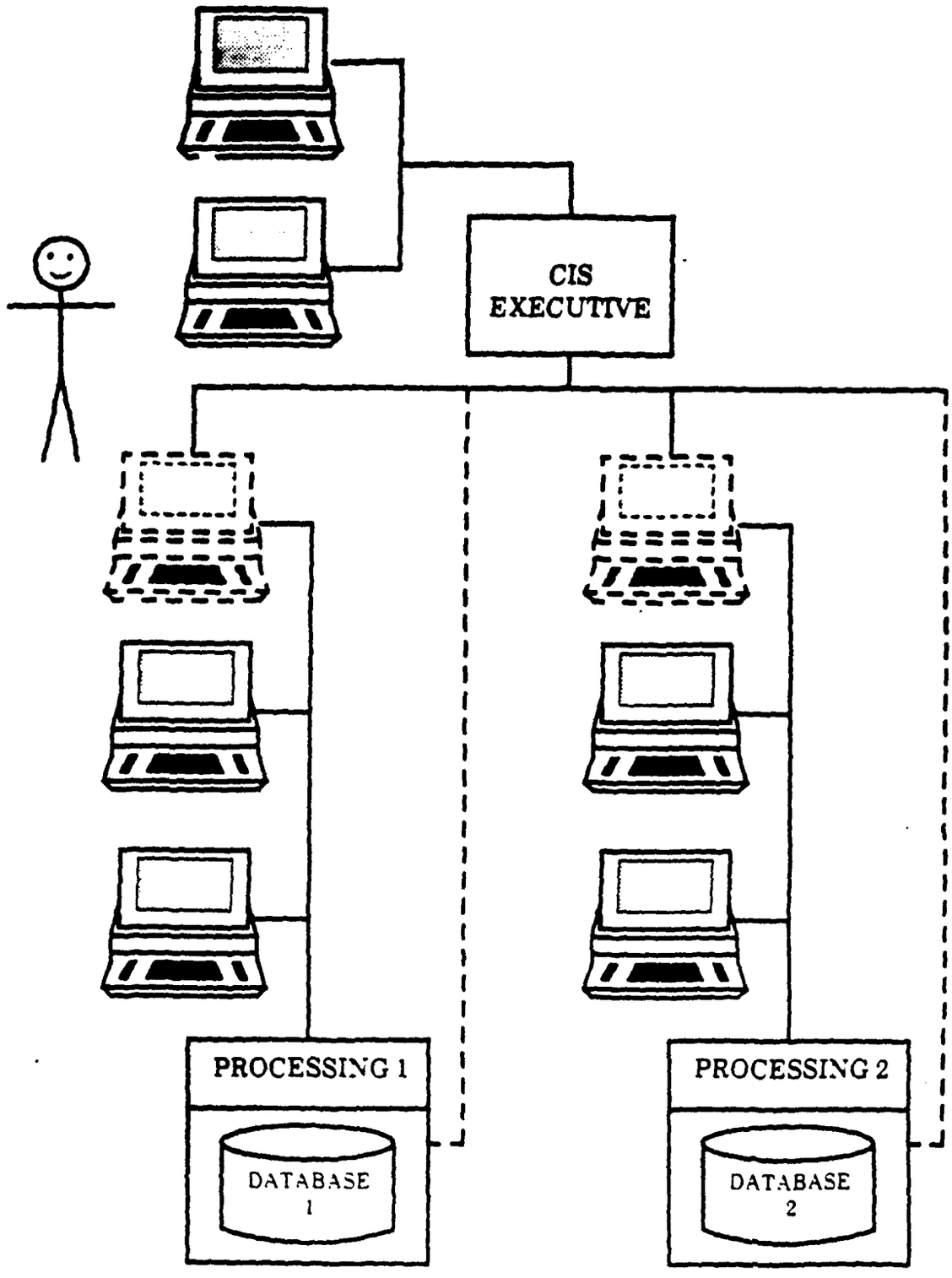


Figure 7. Logical separation of processing from database (Stage 3)

Evolution up to this stage has been, in general, software based. The partitioned processing and database components still run on the same computer. The next stage involves increasing physical separation.

4.4 Physical Separation (Stage 4)

When new computing facility is needed to upgrade the system, two methods are available for partitioning the evolving system: a) migrate a mixture of processing and database components to the new computing facility, and b) partition the processing and database components *physically* and migrate only one type of components (i.e, either the processing components or the database components) to the new computing facility. The second method is advocated for the following reason.

On the one hand, one of the often neglected considerations in planning information systems is the need to operate within an environment of "loosely coupled" organizations. The proliferation of personal computers in most organizations is a manifestation of the desire of individual departments or people to control their own computational destiny. On the other hand, it is being rapidly recognized that databases are important resources and the capability to provide timely access can be crucial.

The method of physical separation of processing and database addresses both of these forces by centralizing the databases onto "file servers" or "database servers" that can be accessed by individually controlled (and "owned") application processing elements --- which may range from personal computers to large-scale mainframes, as illustrated in Figure 8.

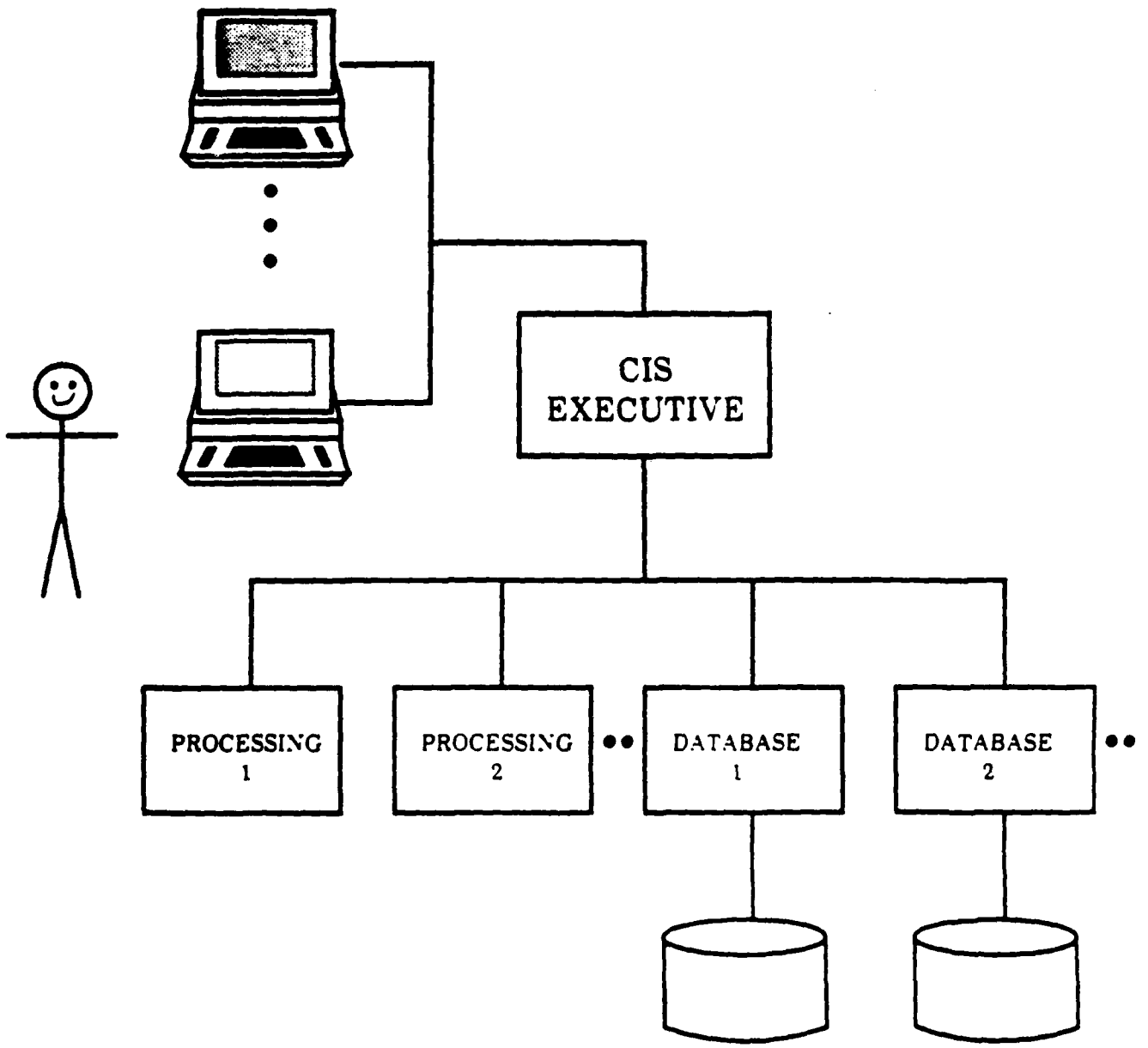


Figure 8. Physical separation of processing from database (Stage 4)

Furthermore, the separation of the application processing from the data processing paves the way for progressing to the specialized functional engine stage.

4.5 Specialized Functional Engines (Stage 5).

The increasing demand for information processing capacity has prompted researchers to design large, cost-effective memory systems with rapid access time. One research direction involves database computers which are computers dedicated and optimized for data management [e.g., 6, 12, 16, 17, 18]. Many of these database computers have adopted highly-parallel, multi-processing architectures to cope with the requirement of high throughput, high reliability, and large storage capacity. Specialization enables the database computer to handle search, retrieval, and storage of large volume of data more effectively, to provide for adequate capacity to perform complicated data restructuring and mapping, and to enforce security and integrity constraints.

Assuming that an organization has progressed to stage 4, as the technology for database computers continues to mature, the organization can easily upgrade system capacity by migrating the database management tasks performed on a conventional computer to a database computer. Meanwhile, proliferation of professional workstations and personal computers will continue to offload many processing tasks currently performed on a centralized computer. A picture of information systems will emerge as depicted in Figure 9.

Many of the tasks performed by the CIS Executive could be migrated to the database management system or the database computer, such as view mapping, data format conversion, and report generation. These features simplify the task of the CIS Executive which now may reside in the professional workstation or

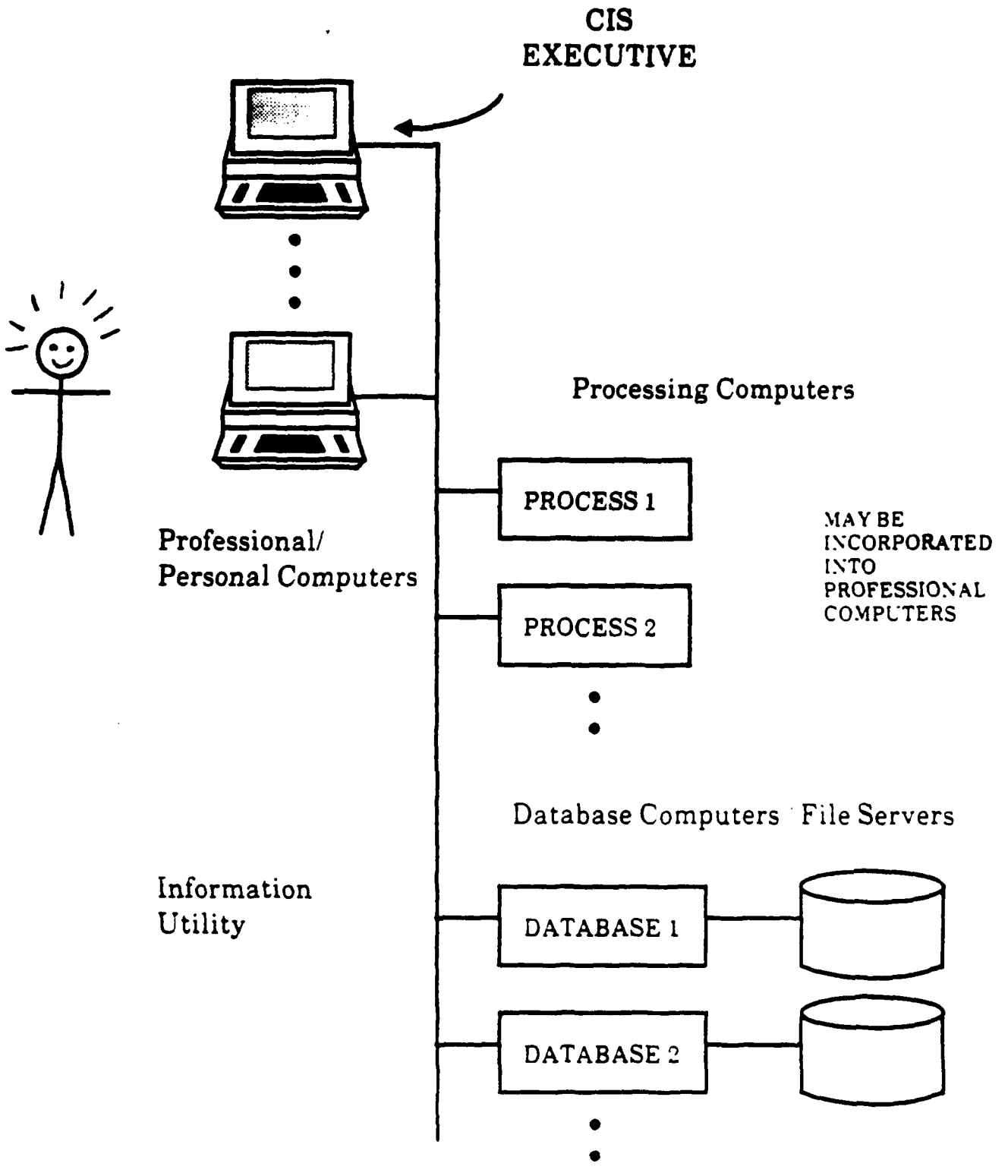


Figure 9. Specialized engines for processing, database and CIS (Stage 5)

personal computer to coordinate access to the resources (processing and database) of the network as well as to mediate steps of internal processing.

The database managed by specialized database computers and controlled by information system specialists of various sub-units of an organization constitute the *information utility* [3, 15]. The end-users, via their desktop computers, access the information utility for data that is either directly usable, usable after further processing by some processing nodes in the network, or usable after further processing by the desktop computer. The CIS at this stage becomes part of the organization's infrastructure to facilitate strategic goals.

5. Concluding Remarks

There are enormous opportunities for businesses to gain competitive advantage through inter-corporate, inter-divisional, inter-product, and inter-model applications. These opportunities for strategic uses of database technology in organizations are often blocked by the difficulties of evolving the existing information technology infrastructure in a rapid, yet non-disruptive manner. The five stage evolutionary process presented in this paper has been found to be effective in overcoming this problem.

The pioneer work on CIS began almost a decade ago [14]. In our recent work [8] , we have found that this architecture is especially effective in an autonomous, evolutionary, and integrative information systems environment. These preliminary results has provided a foundation for the study of even more advanced applications and technologies to support Composite Information Systems.

References

1. Appleton, D.S.,"Very Large Projects," Datamation,January 1986.

2. Benjamin, R.I. et. al. "Information Technology: A Strategic Opportunity," Sloan Management Review, Spring 1985, 25(3), p. 3-10.

3. Baum, R.I. and Hsiao, D.K. "Database Computers - A Step towards Data Utilities," IEEE Transactions on Computers, Vol. C-25, No. 12, December 1976.

4. Cash, J. and Konsynski, B.. "IS Redraws Competitive Boundaries, " Harvard Business Review, March-April, 1985.

5. Clemons, E.R., and McFarlan, E.W., "Telecom: hookup or lose out out," Harvard Business Review, July-August, 1986.

6. Computerworld "Database machine's appeal rising," May 20, 1985, pp. W-2.

7. EDP Analyzer, "The Push for 'Connectivity'," May 1986, Vol. 24, No. 5.

8. Frank, W.F., Madnick, S.E., and Wang, Y.R."A Conceptual Modelfor Integrated Autonomous Processing: An International Bank's Experience," WP# 1866-87, Sloan School of Management, MIT, February 1987.

9. EDP Analyzer "The Push for Connectivity," May 1986, Vol. 24, No. 5.

10. Geist, R. M., and Trivedi, K. S. "Optimal Design of Multilevel Storage Hierarchies," IEEE Transactions on Computers, March 1982.

11. Goyal, A. and Agerwala, T. "Performance Analysis of Future Shared Storage Systems," IBM J.ournal of Research and Development, January 1984.

12. Hsiao, D. K. and Madnick, S. E., "Database Machine Architecture in the Context of Information Technology Evolution," Proceedings of the Third International Conference on VLDB, pp. 63-84, October 6-8, 1977.

13. Keen, P.W., Competing In Time: Using Telecommunications for Competitive Advantage, Ballinger, 1986.

14. Lam, C.Y. and Madnick, S.E., "Composite Information Systems - A New Concept in Information Systems," CISR WP# 35, Sloan School of Management, MIT, May 1978.

15. Madnick, S. E., "Trends in Computers and Computing: The Information Utility," Science, Vol. 185, March 1977, pp. 1191-1199.

16. Madnick, S. E., "The INFOPLEX Database Computer, Concepts and Directions," Proc. IEEE Comp. Con., February 1979, pp. 168-176.

17. Madnick, S. E. and Hsu, "INFOPLEX: Research in a High-Performance Database Computer," IEEE Database Engineering, Vol. 9, No. 1, March 1986.

18. Madnick, S. E. and Wang, Y. R., "Modeling Multiprocessor Computer Systems with Unbalanced Flows," Performance Evaluation Review, Vol. 14, No. 1, May, 1986.

19. Madnick, S.E. and Wang, Y.R., "Key Concerns of Executives Making IS Decisions," WP# 1861-87, Sloan School of Management, MIT, December 1986.

20. Porter, M. Competitive Advantage, Free Press, New York, New York, 1985.

21. Porter, M. and Millar, V.E., "How Information Gives you Competitive Advantages," Harvard Business Review, July-August 1985, p. 149-160.

22. Rockart, J.F. and Scott Morton, M.S., "Implications of Changes in Information Technology for Corporate Strategy," Interfaces, Vol. 14, No. 1, January-February, 1984, pp. 84-95.

23. Trivedi, K. S. and Sigmon, T. M. "Optimal Design of Linear Storage Hierarchies," Journal of ACM, April 1981, pp. 270-288.

24. Wiseman, C. Strategy and Computers: Information Systems as Competitive Weapons, Dow Jones - Irwin, 1985.

DISTRIBUTED HOMOGENEOUS DATABASE SYSTEMS: A COMPARISON BETWEEN ORACLE AND INGRES

ROBERT W. GREF, II

Most large organizations have multiple databases for reasons of *economy* (i.e., smaller computers may be cheaper to own and/or operate than a single large computer), *organization* (i.e., each group wants to operate their own system), and *geography* (i.e., the various groups are geographically distributed and wish to have their databaes nearby for performance, reliability, or security).

In this research effort the concept of distributed homogeneous database systems is explored. By homogeneous we mean that the same, or very similar, database software is used at all sites. Two recent commercial products, Relational Technology's INGRES/Star and ORACLE's SQL*Star, are evaluated and compared. Key areas of concern in the design of distributed homogeneous database systems are *transparency* and *independence*.

By transparency we mean that although the database is distributed, the user should be able to use it as if it were a single database. Six properties of transparency are:

1. Retrieval transparency -- implies that the same results should be obtained regardless of the site from which the retrieval command is executed. Both ORACLE and INGRES meet this requirement.
2. Update transparency -- implies that the user can update the database from any site. Both ORACLE and INGRES provide this capability.
3. Schema transparency -- implies that the result of a schema change command should be visible at all sites, regardless of the site from which it was issued. Only INGRES provides this capability.
4. Performance transparency -- implies that all sites should see comparable performance if the same query is performed, this usually means that the system employs a global optimizer that determines the best site for each operation to be performed and minimizes data transfer. Only INGRES provides this.
5. Transaction transparency -- implies that a single transaction composed of multiple updates is properly and efficiently executed against the correct sites automatically. Neither system currently offers this capability.
6. Copy transparency -- implies that multiple redundant copies of data is supported and the system automatically maintains consistent values and efficiently makes use of these copies to optimize performance and recover from failures. Neither system currently supports this concept.

By independence, we mean that the system should be independent of external factors, such as (1) site crashes, (2) recovery actions, (3) communication networks, (4) hardware and OS, and (5) specific DBMS's used. In these areas of independence ORACLE was found to be more effective at present.

Although significant progress has been made in research on and implementation of distributed homogeneous database systems, full realization of transparency and independence is still a major research challenge.

TECHNICAL REPORT #11

CHAPTER ONE

INTRODUCTION

With the vastly decreasing cost of computers, there is a trend toward greater decentralization of data processing in most companies.[1] Thus, in most organizations today the computing environment is extremely diverse. For example, it is not at all uncommon to find large IBM mainframes, DEC minicomputers, and IBM PCs within a single organization. Within a large company, there are many specialized divisions which solve their problems with a variety of different hardware and software. Naturally, different types of hardware, operating systems and networks lead to incompatibility problems. Thus, "islands of information" develop which make managing corporate resources extremely difficult.

Managers would like to have a global view of all corporate data. However, networking software only helps solve a small fraction of this problem since it is only possible to access one database at a time or upload/download files.[2]

Distributed database technology is the new "hot" area that attempts to solve the "islands of information" problem.

This technology makes it possible for organizations to develop applications and share data across a wide spectrum of machines as easily as if all the information was available on a single computer.[3]

Distributed Processing vs. Distributed Database

Sometimes the terms distributed processing and distributed database are misinterpreted. Distributed processing occurs when programs on different network nodes coordinate with each other by sending messages to one another. If you are a user of a distributed processing environment you must know the database you are trying to access and you must send messages to the other relevant nodes to initiate your process.

In contrast, a distributed database takes care of coordination issues so the user does not have to know where the data is located.[4] Therefore, a distributed database is different from traditional database technology in that it provides a single view for the user to see all the data stored within an entire network of computers.

Problems of Having Information on Separate Computers

There are four major goals of a distributed database system which follow.

First, users realize that there is an enormous amount of data within their organization. What they would like to be able to do is to access the data as if it were a single database.

Second, users want their information to be current.

Third, users would like to be able to access the data without programming. That is, they would like an easy to use language in order to interact with the database.

Finally, users like to control the databases they establish. By maintaining control of "their" data, they insure the integrity of their database is preserved.[5] These four goals lead us to consider what are the advantages of using a Distributed Relational Database Management System.

Advantages of Distributed Databases

There are five major benefits which a distributed RDBMS provides which follow.

First, a distributed database tracks the location of the

data for the user and this provides universal access to the information.

Second, security and integrity are insured at each site. That is, the security is maintained at the local level by the "owner" of the machine rather than at a global level by one database administrator (DBA).

Third, by using distributed systems it is possible to add computers gradually. In other words, it is not necessary to replace all your computers, rather, you can add computers modularly as you need increased capacity/power.

Fourth, data throughput can be increased by two means: (1) dividing up the work among several nodes and (2) greater availability of the data through replication.

Finally, by replicating data you allow some part of the network to fail and yet all the applications that need to access the data will not crash.[6]

Why Would an Organization Purchase a Distributed DBMS?

There are three primary reasons why a firm might purchase a distributed DBMS which are: Economic, Organizational and Technical.

Economic

Currently, many organizations have databases that are so large that they cannot be run on just a single machine. At the same time, the cost of smaller machines has declined significantly making them competitive with large mainframes (traditionally DBMSs have been run on larger mainframes). Finally, the cost of communication equipment to link the computers has also fallen in price.

Organizational

Most organizations that use DBMSs are both geographically and organizationally distributed. It is often the case that the various applications are developed independently but at some point in time there is a need to integrate the systems (perhaps a merger occurs). Thus, in order to be able to model the organization effectively, DBMSs must be distributed.[7]

Technical

Because the data is located close to its users, data availability and performance are greatly improved using a distributed DBMS. Furthermore, if copies of the data are used at different sites the impact of a machine failure is far less.[8]

Thesis Organization

The next chapter is a general chapter on Distributed Relational Database Management Systems and associated terminology. Chapter Three describes ORACLE's Distributed Relational Database Management System. Chapter Four examines INGRES' Distributed RDBMS. Chapter Five is a comparison between ORACLE and INGRES' Distributed RDBMS. Chapter Six steps back from the more technical evaluation and examines what problems these systems solve, what problems they do not solve, and what problems they introduce. Chapter Seven is a mini-case study examining why a major international bank chose distributed INGRES. Chapter Eight is a mini-case study dealing with a major government research Laboratory which uses distributed ORACLE.

CHAPTER TWO

DISTRIBUTED RDBMS AND ASSOCIATED TERMINOLOGY

In this chapter, I explain three terms associated with distributed relational database management systems which are: Transparency Rules, Independence Rules, and Concurrency.

Transparency Rules

Introduction

Both ORACLE and INGRES have released distributed relational database systems. Each vendor means something slightly different by the term transparency, so it would be helpful if some rules could be used to compare these two systems.

As described in Chapter One, a distributed database should allow a user to access data contained in multiple databases at different sites as if he was accessing a single database. The user should not have to know the location of the data he is accessing and this is called transparency.[9]

In order to help explain the six rules that deal with

transparency, let's create an example using Boston, San Fransisco and Dallas as three sites and defining two relations which follow:

```
Boston:         SUPPLIER (sname, status, city, partno)
San Fransisco: PARTNO (ppartno, color, weight)
```

Transparency Rules

There are six rules which are:

(1) Retrieval transparency
(2) Update transparency
(3) Schema transparency
(4) Performance transparency
(5) Transaction transparency
(6) Copy transparency

Retrieval Transparency

Retrieval transparency means the same results should be obtained regardless of the site in the distributed network where the retrieval command is executed. Thus, if the following example is run, the results should be the same regardless if executed in Boston, San Fransisco, or Dallas.

```
select sname
from SUPPLIER
where partno in
    select ppartno
    from PARTNO
    where weight = 100
```

Update Transparency

This rule states that the update should not restrict the user to updating (i.e. delete, insert, replace) at a

particular site. Thus, regardless of which site the user executes an update the result of the following example should be correct:[10]

```
update SUPPLIER
set status = in stock
where partno in
    select ppartno
    from PARTNO
    where weight = 100
```

Schema Transparency

The idea of schema transparency is that regardless of which site you are at in the distributed system, the result of the command is visible at all sites.

Two examples of this concept follow:

(1) Suppose we create an ALIAS for a relation:

alias for SUPPLIER at San Fransisco is SUPP

Thus, this alias SUPP must be valid in the distributed database system at all 3 sites with a single command.

(2) Suppose we issue a CREATE command in San Fransisco:

create PART (name = part3, description = tool)

Again this PART relation must be visible at Boston, San Fransisco and Dallas using a single command.

Thus, what is necessary to have schema transparency is a coordinated data dictionary. If the distributed database system did not have a true data dictionary, it would fail the test because it would be necessary to execute the schema

command at each site.

Performance Transparency

Performance transparency states that performance should be comparable regardless from which site the command is executed.

The only way to achieve consistent performance results is to utilize a distributed query optimizer. Thus, regardless of where the command is executed, the access plan will be independent and hence comparable performance will result (except for transmission delays).[11]

Example:

```
select suppliername
from SUPPLIER
where partno in
   select ppartno
   from PARTNO
   where weight = 100
```

In this case, it is likely that the query optimizer will first choose to execute the inner block of code at Boston and then move the result to San Fransisco for additional processing. In order to illustrate the situation without a distributed query optimizer, let us consider the above example executed at Dallas (without a distributed optimizer). In this case, the entire relation SUPPLIER as well as part of PARTNO would be moved to Dallas before any processing of the command took place. Therefore, you can see that the performance results could vary substantially without

a distributed query optimizer because in some cases performance could virtually mimic that of the distributed query optimizer, whereas in the above case, there could be substantially more variation. The conclusion one should draw is that without a distributed query optimizer, the performance depends on which site the query was executed at.

Transaction Transparency

The transaction transparency rule states that when a transaction (which may consist of many query language commands) contains multiple updates, the transaction must be able to be executed by the distributed database system. Basically, the system should act like one-site transaction systems. That is, a one-site transaction system requires that a transaction must be serialized in order to avoid problems with other concurrent transactions.

Copy Transparency

The copy transparency rule states that a distributed database should support multiple copies of objects so that a high availability is achieved for the user which results in better retrieval performance. Thus, if the system crashes, a redundant copy can be used until the failure is repaired. It is the responsibility of the data manager to set up some scheme to address the problems associated with redundant copies, such as: (1) how to update multiple copies, and (2) how to restore operation after a system crash.[12]

Independence

Introduction

Independence Rules like Transparency Rules should be used to help judge the "goodness" of a distributed database system.

There are five rules which are:

(1) Crash Independence
(2) Recovery Indepedence
(3) Network Independence
(4) Hardware/OS Independence
(5) DBMS Independence

Crash Independence

The crash independence rule states that if a crash occurs in a distributed database system, that it should effect the availability of data only for data residing at that node. For example, if machine A crashes, machine B should be unaffected unless it needs to access data from machine A.[13]

Recovery Independence

This rule states that if a system crashes, it should be able to recover automatically.

Network Independence

The network independence rule states that the

distributed database system can operate across any networks that are available. That is, it should be possible for the user to develop custom protocols so that even networks not directly supported by the DBMS vendor can be used.

Hardware/OS Independence

This rule means that the distributed DBMS should be available for a wide range of hardware (from PCs to mainframes) and operating systems.[14]

DBMS Independence

The DBMS independence rule means that the distributed DBMS should support other common DBMS software, such as DB2 or SQL/DS.[15]

Concurrency Control

Introduction

Both ORACLE and INGRES are multi-transaction systems. In a multi-transaction system many transactions may be happening concurrently. Thus, it is possible that one transaction might interfere with each other and therefore, it is useful to look at how a distributed database system handles concurrency issues.[16]

Locking

In a distributed system, many transactions may be taking

place at the same time and thus, it is possible for one transaction to interfere with another. For example, imagine that an update is issued to record #1 while at the same time a transaction is retrieving record #1. A mechanism to prevent such a situation from occurring is called locking. Thus, one transaction acquires a lock on a particular action so that another transaction cannot occur until the lock is released.[17]

Locks solve a great many problems but also create problems as well. For example: Suppose a transaction #1 must acquire two different locks, A and B. Let's assume lock B is not yet available, so transaction #1 locks A and waits for B to be released. At the same time, transaction #2 does not realize that transaction #1 has locked B and is waiting for lock A to be released. This situation is called a deadlock since both transactions are waiting for each other to complete.

Introduction to Figure 1

Figure 1 is a comparison between distributed INGRES and distributed ORACLE based on thirteen categories which I feel are important when considering a distributed database system.

In Chapter Three, I go into detail on the items mentioned in the INGRES column of the table. In Chapter

Four, I expound more fully on the ORACLE column of the table. Then, in Chapter Five, I give my opinion on each of the thirteen points.

INGRES VS. ORACLE - Figure 1.

FEATURE/FUNCTION	INGRES	ORACLE	EVALUATION
(1) Transparency Rules			INGRES is better as far as transparency rules are concerned.
- Retrieval	Yes	Yes	
- Update	Yes	Yes	
- Schema	Yes	No	
- Performance	Yes	No	ORACLE is better with regard to indepedence rules.
- Transaction	No	No	
- Copy	No	No	
Independence Rules			
- Crash	Fair	Excellent	
- Recovery	Fair	Excellent	
- Network	Supports only 2	Supports Many	
- Hardware/OS	Fair	Excellent	
- DBMS	Fair	Excellent	
(2) Concurrency Control	Readers block writers & writers block readers.	Readers do not block writers & writers do not block readers.	My personal preference is record level locking since it is more specific and would result in fewer deadlock situations.
	INGRES uses page level locking.	ORACLE uses record level locking.	
	Enabling read consistency increases lock frequency. Thus, sacrifices concurrency.	ORACLE provides read consistency without increasing lock frequency.	
(3) Query Optimizer Features	INGRES has an elaborate query optimizer based on statistics.	ORACLE uses A.I. to help aid in querying.	Both databases offer good query optimizers.
	INGRES sees their query optimizer as a major strength.		
(4) Use of Personal Computers	INGRES has recently released PC version.	PC is very integrated in	INGRES PC version 5.0 is faster than

INGRES VS. ORACLE - Figure 1.

FEATURE/FUNCTION	INGRES	ORACLE	EVALUATION
	Have choice of 2 different interfaces you can use: regular INGRES or PCLINK. In benchmark tests, INGRES 5.0 outperformed ORACLE 4.0. Only have B-TREE file access on PC.	ORACLE. Can join tables on PC with tables on host in ORACLE. PC Port is fully functional ORACLE.	ORACLE version 4.0. ORACLE seems to offer a more fully functional version.
(5) Compatibility	INGRES claims to be fully ANSI SQL compatible.	ORACLE made the decision early on to be ANSI SQL compatible.	ORACLE seems to offer more here since it has been fully SQL compatible since it began.
(6) Portability	Runs on IBM VM/CMS, DEC VAX/VMS, IBM PC, 2 dozen Unix machines.	ORACLE available on IBM VM/CMS, Various Unix, IBM PCs, IBM MVS/SP, DG AOS/VS, VAX/VMS, Apollo Domain.	ORACLE seems to have the advantage here - it is more portable and runs on more systems than INGRES.
(7) VMS System Interface	INGRES has UNIX origins - badly ported. Requires 2 tasks for each user (due to pipes). INGRES does not have a shared buffer pool and no shared code. Uses native data types. Thus, need to execute far less instruction sets.	ORACLE takes advantage of variety of VMS system utilities. Example: ORACLE has shared buffer pool and shared code. Does not use native data types.	Both systems are good. ORACLE does take advantage of a variety of VMS facilities and utilities and achieves good performance. However, INGRES uses native data types - more highly tuned in this regard. Also, makes use of VMS distributed lock manager.

INGRES VS. ORACLE - Figure 1.

FEATURE/FUNCTION	INGRES	ORACLE	EVALUATION
	VMS distributed lock manager built into product.	Does not use VMS distributed lock manager.	
(8) IBM VM/CMS	INGRES only recently entered this market. INGRES uses minidisks concept.	ORACLE uses shared global area (SGA) to maximize performance However, security may be an issue under IBM VM/CMS.	INGRES is slower but more secure than ORACLE under IBM VM/CMS.
(9) Performance Related Issues	INGRES has file access methods (regular and compressed versions): ISAM, HASH, HEAP, SORTED HEAP, and B-TREE.	ORACLE uses only B-TREE file access method.	I tend to believe ORACLE is more performance oriented than INGRES, however, INGRES does have more file access methods.
	Can use only B-TREE method on PCs.	Supports nulls - can differentiate between nulls, blanks and zero values.	
	Does not support nulls.		
	Limits tables to 127 columns and rows to 2,008 bytes & does not support long text.	Relation can have unlimited number of rows and up to 255 columns.	
		ORACLE supports long text.	
	INGRES allows you to copy records in batch.	Has an array interface which allows to copy records in batch.	
		ORACLE has an internal sort.	
	INGRES can use only one index to process a query.	ORACLE can use multiple indexes to process a query.	

INGRES VS. ORACLE - Figure 1.

FEATURE/FUNCTION	INGRES	ORACLE	EVALUATION
(10) Query Language	SQL QUEL INGRES was not ANSI SQL compatible. They claim they are now. Restricition (inability) to handle VAX packed decimal format.	SQL ORACLE made the strategic decision early on to be ANSI SQL compatible. ORACLE SQL has significant, powerful extensions to ANSI SQL.	Many users do find QUEL to be more powerful than SQL. SQL has been adopted by: ANSI, the database industry, and database users as the standard query language. I believe INGRES SQL has several idiosyncrasies. Example: no null support.
(11) 4 GL Tools	Powerful, fast & easy to use set of tools. INGRES users really like the 4 GL Tools.	ORACLE has a variety of tools available.	Both systems offer good tools. Some users would like to be able to use a more user-friendly interface similiar to the MacIntosh.
(12) Security	Does offer a security audit facility. Table-by-table journaling. Audit log right in dictionary.	Provides a security audit facility. ORACLE allows you to have view isolation and protected tables.	Overall, ORACLE seems better in this regard.
(13) Network Support	INGRES supports DECNET and TCP/IP.	ORACLE supports DECNET, TCP/IP, RS 232 asynchronous TTY communications, 3270 PC/Irma Protocol, EtherNet LAN with TCP/IP, and ORACLE supports	ORACLE is superior in this regard. Supports much larger variety than INGRES.

INGRES VS. ORACLE - Figure 1.

FEATURE/FUNCTION	INGRES	ORACLE	EVALUATION
		customers who want to develop custom protocols.	

CHAPTER THREE

ORACLE DISTRIBUTED RDBMS

Background On ORACLE Corporation

ORACLE was founded in 1977 and in 1979 they introduced a commercial relational database management system (RDBMS).[18] ORACLE introduced their RDBMS based on IBM's Structured Query Language (SQL) database language three years before IBM released its SQL/DS in 1982 and DB2 in 1985.[19] Since 1979 it has been installed in 4,000 sites worldwide and over 10,000 personal computer copies have been sold. Since ORACLE was developed using the language C and all versions are ported from the same source code, it is easy to move ORACLE among different systems.[20] Thus, ORACLE can be run on a wide variety of mainframes, minicomputers and microcomputers.

ORACLE has recently increased the performance of their relational DBMS significantly as illustrated by an up to 20:1 increase in the performance of certain queries in Version 5 compared to Version 4. This increase was achieved primarily through the following three product enhancements: (1) significant performance improvement in OR operator processing, (2) a faster sort/merge routine for processing join and group by queries, and (3) a new array interface

which allows transferring many rows at a time rather than just a record at a time. Thus, these three new features speed query processing. In addition, the array interface and the sort/merge routine increase the transaction processing performance.[21]

ORACLE offers far more than just a relational DBMS. ORACLE offers several interfaces from which a user can work with the database, ranging from one intended for novice users to an advanced programming interface for MIS professionals. Regardless of the interface chosen, the end result is that ORACLE produces SQL queries which are processed by ORACLE. ORACLE offers a variety of other options such as SQL*Forms, an ad hoc data and reporting tool. Another option offered is SQL*Net which provides network communications. ORACLE also supports language interfaces for Cobol, Fortran, Basic, Pascal, PLI and Ada.[22]

In 1986, Oracle Corporation announced SQL*Star, a distributed relational DBMS. SQL*Star makes it possible to distribute databases among both ORACLE and non-ORACLE DBMSs. The foundation of the SQL*Star package is SQL*Net along with the protocol set.

General Advantages of ORACLE

First, ORACLE has a fully functional microcomputer version of their system. The personal computer version of ORACLE requires 512 K, a hard disk, and DOS 2.0 or later.[23] This is a great advantage since it is possible to have a common system run on mainframes, minicomputers, and microcomputers. Thus, it becomes possible for a user to develop an application on a micro and run it on a minicomputer or mainframe.[24]

Second, the user and the vendor benefit from the fact that ORACLE was written in C. This makes it quite easy for ORACLE to work on a variety of hardware. Therefore, ORACLE is available to a large number of users.

Third, users benefit by ORACLE's standard ANSI structured query language (SQL). ORACLE is plug compatible with SQL/DS and DB2 and they offer additional benefits over SQL/DS and DB2, such as portability.

Fourth, a major benefit provided by ORACLE is the introduction of SQL*Star.[25]

SQL*Star

SQL*Star is a family of products which include SQL*Connect and SQL*Net products which form a distributed

RDBMS. SQL*Star allows you to create applications on one computer and access your data from other computers (see Figure 2). Thus, SQL*Star gives the user the impression that he is dealing with a single database. SQL*Star has what is known as an open architecture and thus allows mainframes, minicomputers and personal computers to be tied together saving on storage costs as well as providing easy to use interfaces.

SQL*Star provides three types of independence that make it easy to use, which are: location independence, network independence and DBMS independence.

Location Independence

This feature allows the user to perceive the data as existing in a single database. That is, SQL*Star takes care of locating your data and figuring out how to get it. For example, your data could be spread among many different machines at many different locations and SQL*Star would take care of finding it and retrieving it (see Figure 3).

Network Independence

Network independence is provided by SQL*Star which allows you to access your data from any computer in your organization's computer network.

DBMS Independence

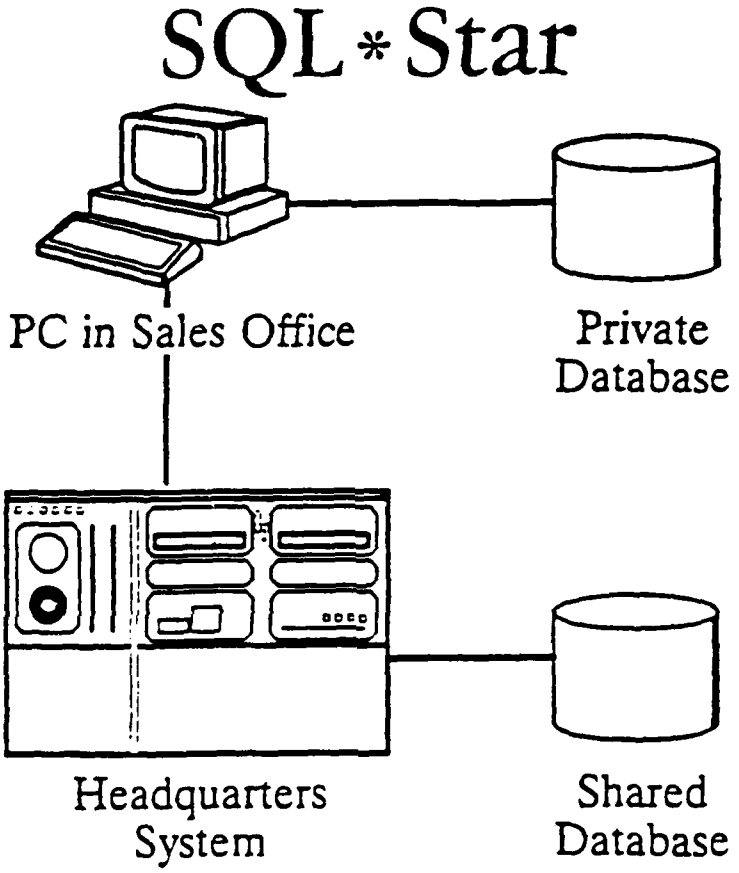

SQL*Star
PC in Sales Office
Private Database
Headquarters System
Shared Database

Figure 2

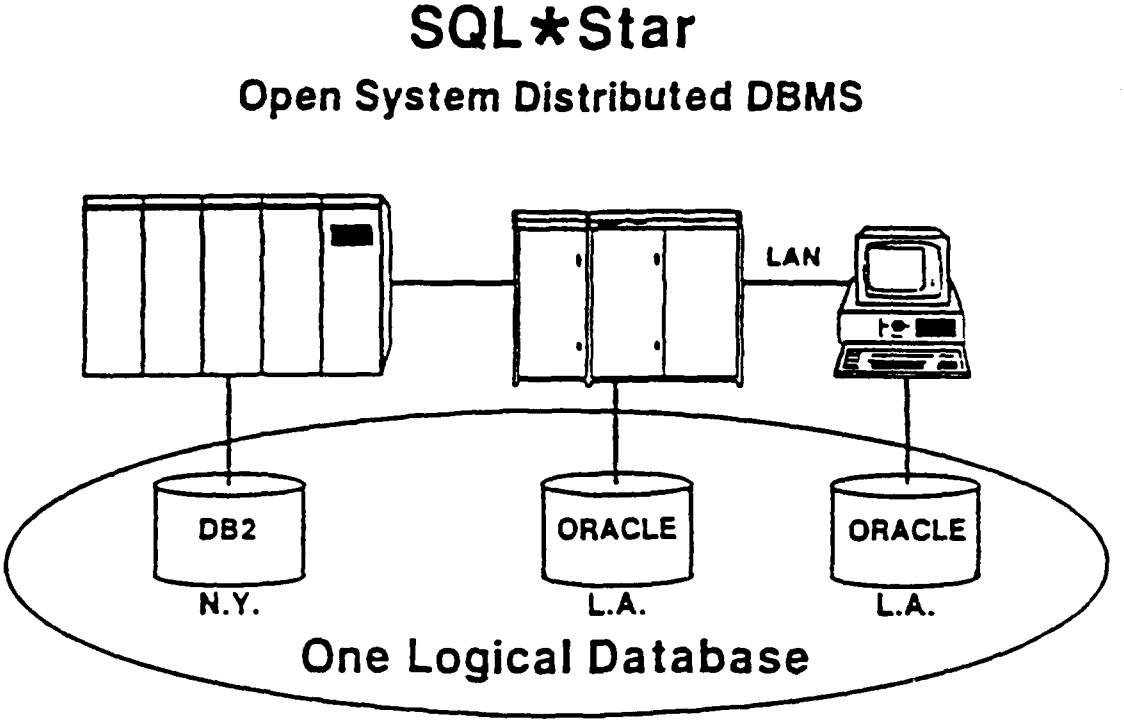

SQL*Star
Open System Distributed DBMS
LAN
DB2
N.Y.
ORACLE
L.A.
ORACLE
L.A.
One Logical Database

Figure 3

SQL*Star's open system architecture allows the system to be portable. Thus, it is possible to distribute your data not only on ORACLE databases, but also on DB2 and SQL/DS databases.[26]

At the heart of SQL*Star is SQL*Net which is described below.

SQL*Net

What is SQL*Net?

ORACLE's SQL*Net Users Guide says, "SQL*Net allows applications to reside on a machine other than where the database is located, and provides a means of moving data from one node to another on the network" (see Figure 4). It is possible for ORACLE applications to retrieve data from a remote location using SQL*Net. Also, one can with a single SQL statement reference multiple nodes and do joins across the network.[27]

SQL*Net Architecture

SQL*Net is able to do distributed processing by using both multi-node network communication and process-to-process communications. Thus, ORACLE uses a two-task architecture which means that the user task is a separate process from the server task. The user task and the server task communicate

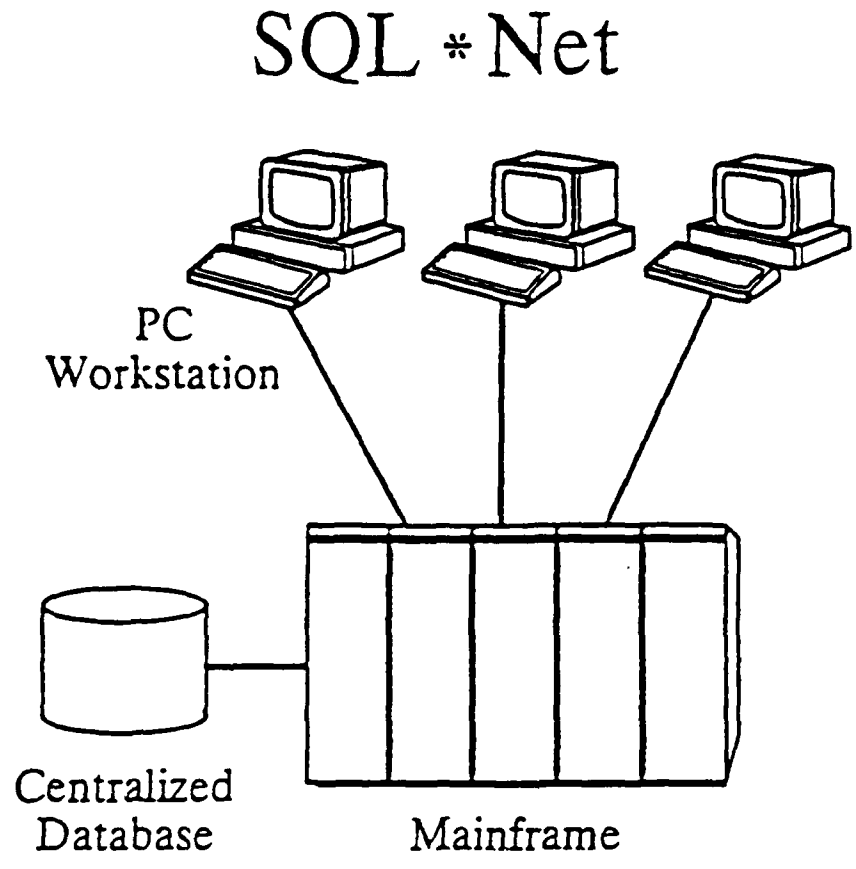


Figure 4

with one another [via interprocess communication support by the operating system] to complete a single job (see Figure 5).[28] Even when ORACLE runs on a single machine, the two-task architecture is maintained by treating the application as a front end process and the ORACLE kernel functions as the back end process. If two processes are running on two different machines, then they communicate with one another via communication protocols supported by SQL*Net. One task is the server and the other task is the client. The client is the application program and the server is the ORACLE kernel.

Under ORACLE's terminology, the host is the machine where the database resides and runs the ORACLE kernel thus supporting the server. The machine where the application resides is called the client. However, it is possible for a machine (example: a VAX) to be both a client and a server. In the case of a single user system (example: a PC), it can only be a client (see Figure 6). Please note that it is possible to put an application as well as the application database on a PC but still look up data on a mainframe using ORACLE. Because the PC is a single user system, it is not possible to query your database on the PC and at the same time have someone dial into your PC to query your database. However, under Xenix which allows a multi-user system, it is possible to have the PC be a server. The following table illustrates the various possibilities:[29]

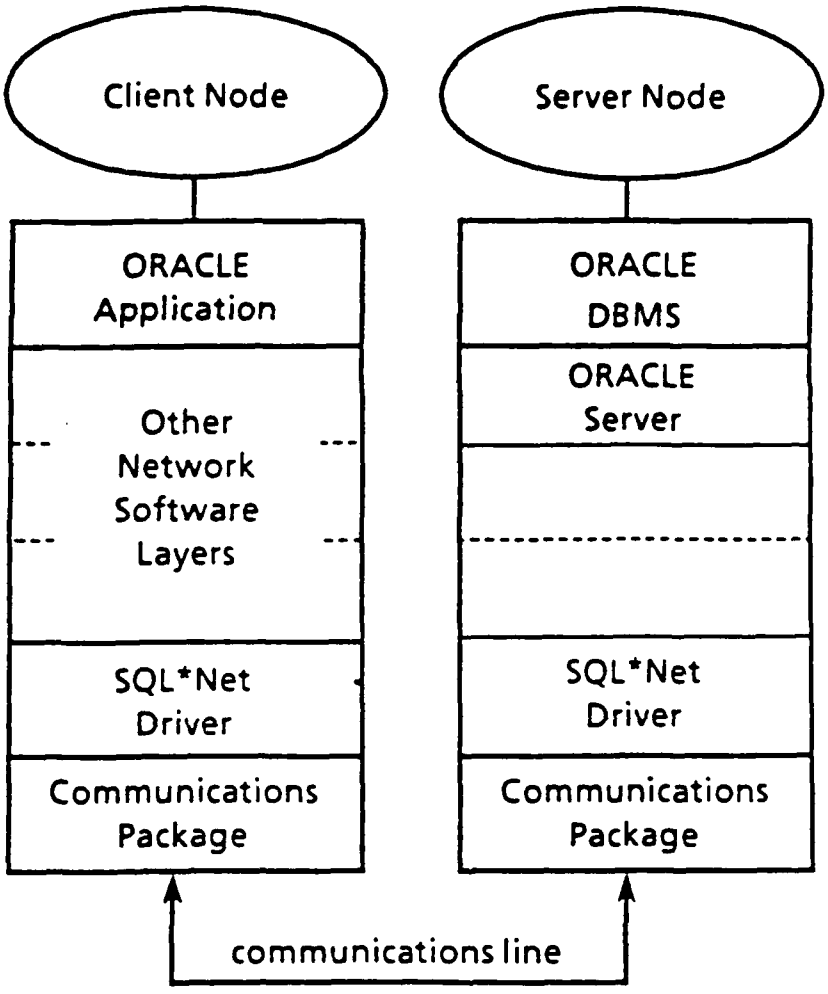

Client Node
Server Node
ORACLE Application
Other Network Software Layers
SQL*Net Driver
Communications Package
ORACLE DBMS
ORACLE Server
SQL*Net Driver
Communications Package
communications line

Figure 5

ENVIRONMENTS SUPPORTED BY SQL*NET

Client	Server	Protocol
PC/MSDOS	VAX/VMS	Asynchronous
PC/MSDOS	DG/AOS/VS	Asynchronous
PC/MSDOS	Various Unix	Asynchronous
PC/MSDOS	IBM/VM/CMS	Asynch, 3270
PC/MSDOS	IBM MVS/SP	Asynch, 3270
Various Unix	Various Unix	TCP/IP
VAX/VMS	VAX/VMS	DECNet

Figure 6

Machine	Client	Server
Personal Computer	Always	Only under Xenix
Minicomputers	Yes	Yes
Mainframe	Possible	Usually

Restrictions of SQL*Net

There are two limitations of using SQL*Net.

First, a single transaction can only update a single node. What I mean by this is that while an application may query several nodes, a single transaction requiring an update can affect only one node. For example, one SQL statement can refer to only one database (think of 1 SQL statement = 1 transaction). For example, it is not possible to issue the command: update all parts by ten percent in a single SQL statement if parts is found on several databases.

Second, SQL*Net does not support distributed transactions due to the fact that each transaction can affect only one node. Therefore, although an application may update several different nodes, the commits and rollbacks for each site are done independently.

Advantages of SQL*Star

In addition to the five advantages of a distributed database that were mentioned in Chapter One (see Advantages of Distributed Databases), SQL*Star also provides the following two advantages:

First, since ORACLE's distributed database system uses the site autonomy approach rather than INGRES's central dictionary method, it is likely that ORACLE will be compatible with IBM's directions in distributed databases.

Second, the site autonomy approach has the advantage that it is not dependant on a central node should the system crash. INGRES/STAR is dependant on a centralized data dictionary and thus it the particular machine that held the data dictionary crashed, all distributed access would be impossible.

Concurrency Control

In ORACLE, readers do not block writers and writers do not block readers.

ORACLE LOCKING

ORACLE uses record level locking. Record level locking

has advantages associated with it as well.

First, it seems to offer higher security. For instance, when reading, the read level locks are not shared. Thus, the locking is done at a finer level and is much more specific than page level locking.

Second, there is a problem under page level locking when users access the same page and one person wants to update this same page. This can result in deadlocks, which take time to detect and hence lower performance in these cases.

There are two strategies for dealing with deadlocks. First, you could decide to prevent deadlocks before they occur. But this strategy requires much locking and releasing of locks which causes a degradation of performance. Second, you could allow deadlocks to happen since they occur infrequently, but have an automatic deadlock detection and recovery scheme. This is the method chosen by INGRES and ORACLE.[30]

Query Optimizer Features

ORACLE has incorporated artificial intelligence in Version 5.0 to help aid in querying. This query optimizer

allows the system to find an efficient method of moving around a database. The way the query optimizer works is that it tries to minimize the number of retrieval requests and utilizes all available indexes in order to proceed as efficiently as possible.

Also, ORACLE's method of SQL allows users to update tables with data retrieved directly by a query, thus enhancing query performance. This is accomplished with the SET clause by nesting queries with an UPDATE command. Thus, the SET command permits users to join the result of several queries into a single result. Also, ORACLE supports MINUS and INTERSECT operators in addition to the UNION operator (which is in DB2). The INTERSECT operator retrieves only those rows which are common to the two query results. MINUS operates by returning the result of these rows retrieved by the first query which are not found in the second query.[31]

Use of Personal Computers

ORACLE provides a fully functional PC version which requires 512K RAM, a hard disk drive, and DOS 2.0 or later. ORACLE's PC product is fully compatible with IBM mainframes and SQL/DS and DB2 systems.[32] Thus, the ORACLE system is especially good if it is necessary to interface within SQL with minicomputer or mainframes.[33]

Compatibility

ORACLE made the decision early on to use IBM's structured query language as its user interface. This proved to be an extremely good decision as in 1986 SQL won wide acceptance in the marketplace.

Portability

ORACLE's SQL*Star is available for: IBM VM/CMS, various Unix, IBM PC's, IBM MVS/SP, DG AOS/VS. ORACLE provides excellent portability by writing their system in C, thus minimizing the cost of transporting to other machines.

VMS System Interfaces

ORACLE takes advantage of a variety of VAX/VMS facilities and utilities to enhance its performance. For example, ORACLE uses a shared buffer pool in order to avoid an excessive input/output load.

On the downside, ORACLE does not use the native data types of the VMS system because ORACLE strives for portability of their system and thus they use their own data types, thus, ORACLE must execute more instructions.

IBM VM/CMS

ORACLE maximizes their performance under the IBM VM/CMS operating system but it is said that they use the shared global area to do so. By using the shared global area, this can cause security problems. INGRES entered this market much later than ORACLE and uses the concept of minidisks which is a multi-file architecture.[34]

Performance And Feature Related Issues

ORACLE does support nulls, that is, it differentiates between nulls, blanks and zero values. Furthermore, a relation can have an unlimited number of rows and up to 255 columns (and therefore can support long text). ORACLE also has an array interface which allows records to be copied in batch which improves performance greatly over copying records one by one. Finally, ORACLE offers an internal sort on frequently performed SQL query options, namely GROUP BY, ORDER BY, CREATE INDEX, and SELECT DISTINCT, which greatly increases sort speed performance.[35]

ORACLE supports only the B-TREE file access mode which is the file mode best suited to most situations. However, it is not possible to use other types of file access modes.

Query Language

ORACLE made the decision in 1981 to use IBM's Structured Query Language as its data language. This became the "official standard" in 1986. Thus, ORACLE is ANSI SQL compatible and has the ability to run DB2 programs without performing any modifications.

4 GL Tools

Many users did like ORACLE's 4 GL tools, especially SQL*Calc, the Forms options, the high-level language interface and the Data Load facility. On the downside, some users felt the report writer facilities were not good enough because they lacked flexibility.[36]

Security

ORACLE offers extensive security features. For example, ORACLE has locking techniques which prevent the users from altering data, called an exclusive lock. Also, they have a locking mechanism which prevents changes to data being read, called a shared lock. Each data owner can use GRANT/REVOKE

statements in order to define security any of the following levels: tables and rows, on a field, or on a field-by-value level. Furthermore, the AUDIT statement can be used to check for unauthorized use, making it possible to track down violators.[37]

Network Support

ORACLE supports a great variety of networks including: DECNET, TCP/IP, RS 232 asynchronous TTY communications, 3270 PC/Irma Protocol, EtherNet LAN with TCP/IP, and ORACLE supports customers who want to develop custom protocols.[38]

CHAPTER FOUR

INGRES DISTRIBUTED RDBMS

Background on INGRES

The INGRES Project was initiated by Michael Stonebraker and Eugene Wong in 1973 at the University of California, Berkley. The goal of the project was to develop a working relational database system.[39] In 1979, Eugene Wong and the other authors of INGRES decided to form a company to produce INGRES as a commercial product. So in October 1980, Relational Technology Inc. (RTI) was founded.[40]

In the past, INGRES concentrated on the minicomputer relational DBMS marketplace, namely, Digital Equipment Corporation's VAX line of computers and UNIX OS. INGRES had about 45 percent of the minicomputer DBMS market. In other words Relational Technology Inc. focused on the scientific and engineering market. In contrast, ORACLE focused primarily on the commercial business market. However, currently they are not so specialized and compete in all areas.

Today, INGRES has perhaps the largest number of Unix-based DBMS customers with approximately 20 percent of

its user base operating in Unix. Overall, the total number of users of INGRES as of December 1986 was approximately 3,750 versus the 4,000 users of ORACLE.[41]

Relational Technology had trouble in the past competing with ORACLE because the INGRES RDBMS was designed for specific operating systems, whereas ORACLE ports their system to a variety of different machines. Although INGRES is also written in C, they develop and maintain separate code for each operating system. In contrast, ORACLE ports are derived from the same source code which is much faster.[42]

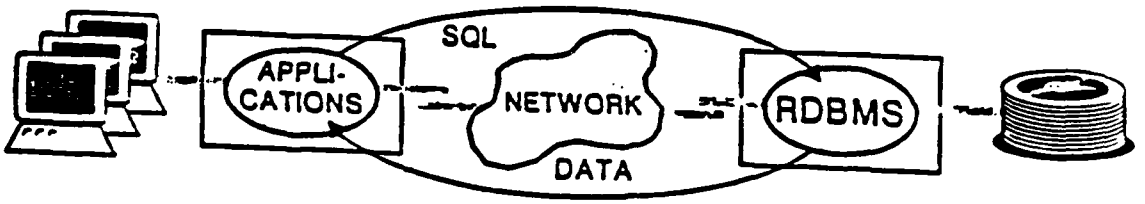
INGRES has extended its RDBMS to distributed database systems by introducing INGRES/Star in 1986. INGRES/Star is built upon INGRES/Net (see Figure 7), the first data networking product, which was introduced in 1983.[43]

INGRES/Star

INGRES/Star is an open-architecture distributed database system which supports a variety of environments, allows a user to transparently access data and ensures high performance and reliability. INGRES/Star consists of an open-architecture distributed database, a set of integrated tools, and high performance SQL (see Figure 8).

According to the INGRES/Star brochure, "INGRES/Star is built upon the INGRES/Net product to translate messages

Distributed Access Networking
INGRES/NET



- Transparent remote access
- Distributed processing
- Multiple network protocols

Figure 7

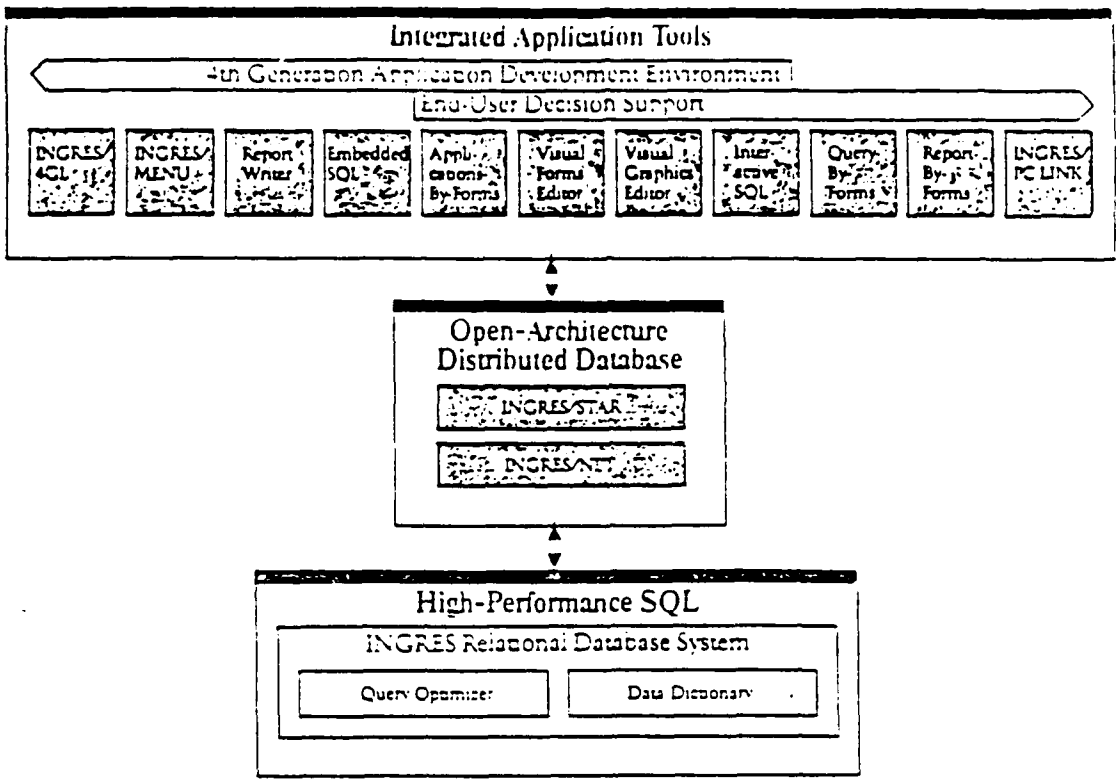

Integrated Application Tools
4th Generation Application Development Environment
End-User Decision Support
INGRES/4GL
INGRES/MENU
Report Writer
Embedded SQL
Visual Forms Editor
Visual Graphics Editor
Interactive SQL
Query-By-Forms
Report-By-Forms
INGRES/PC LINK
Open-Architecture Distributed Database
INGRES/STAR
INGRES/NET
High-Performance SQL
INGRES Relational Database System
Query Optimizer
Data Dictionary

Figure 8

transmitted over the network invisibly, hiding the differences between the disparate machine architectures through a common interface" (see Figure 9). Applications in INGRES communicate with the INGRES relational database manager using a common language which is independent of its environment.

Each machine in the network runs a standard version of INGRES, with each site containing its own local data dictionary. One important feature of INGRES is that it keeps the front-end processes separated from the back-end management software, similar to ORACLE. Applications use SQL in order to ask for data. INGRES uses a distributed database manager which segments and routes SQL queries to the appropriate databases in the network. The distributed database manager in effect coordinates the front-end applications and the back-end databases (see Figure 10).

Features of First Release of INGRES/Star

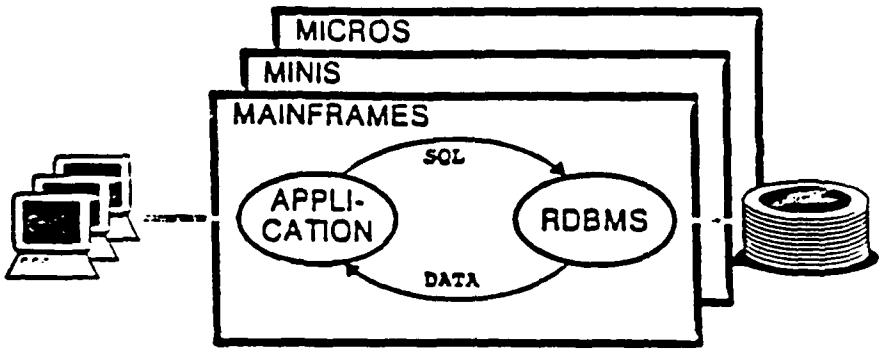
Transparent Retrieval

This feature allows users to access data from multiple sites as if they were querying a single database.

Distributed Transactions with Single-Site Update

Similar to SQL/Star, INGRES/Star allows the users to

INGRES-Portable Applications & DBMS



- Dissimilar hardware
- Dissimilar operating systems
- Dissimilar terminals

Figure 9

Distributed Database Manager
INGRES/STAR

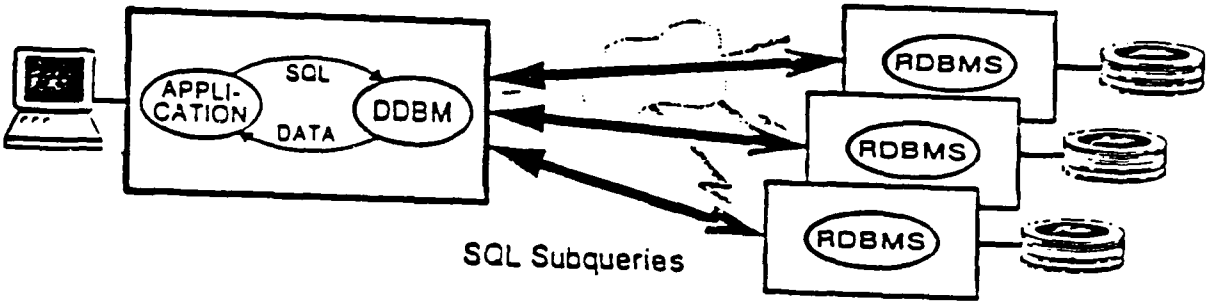


Figure 10

access data from multiple sites in a single query. Any particular transaction which requires an update may only affect one site, but this site may be anywhere in the distributed database network. It is possible, however, for different transactions within a single application program to update multiple sites. Thus, you will notice this is identical to how ORACLE deals with transactions.

Support for Unix and VMS

Currently, INGRES supports most Unix environments as well as DEC VMS environments.

Query Optimizing

INGRES/Star uses a query optimizer to fine tune the system. In the first release, the query optimizer maintains excellent performance by minimizing communications traffic. The INGRES query optimizer is based on statistics pertaining to the database queried. Thus, it chooses a certain access strategy based on the number of occurences in the database of a particular item. For example, say you want to do a join on two tables where: Age>80 and Wage> $1. Thus, the query optimizer would first pull out all employees who were greater than 80 years old first, since almost everyone's salary would be greater than $1.[44]

Integrated Tools

INGRES/Star provides the user with a set of integrated tools which greatly speed development time of applications. Programmers can use 4GL, visual forms editing and host language interfaces including ADA, BASIC, C, COBOL, FORTRAN, PASCAL, and PL/I. INGRES also offers Visual Programming which is used for ad-hoc queries and simple reports which aids in decision support at the end-user level.[45]

Benefits of INGRES/Star

Reduced Costs

INGRES/Star provides application portability, although it does not run on the variety of hardware ORACLE supports. By portability I mean it is possible to develop an application on a microcomputer and run it unchanged on a mini or mainframe computer. This feature means it costs less to develop and maintain an application for the following two reasons: (1) applications can be run on hardware that is most cost-effective and (2) any maintenance changes need be made to only one set of source code.

Also, INGRES/Star allows modular expansion of the network. Thus, it is possible to utilize the current investment in hardware and networks and expand as needed.

Improved Productivity

Application programmers and end-users have the ability

to access data throughout the network.[46] Furthermore, users need only learn a single language (SQL or QUEL) or users can utilize a menu-driven application called INGRES/QBF to interact with the database.[47]

Finally, applications are data independent in the sense that the user does not have to worry about where the data is located or how to get to the desired data.[48]

Higher Performance

By utilizing the data replication feature, it is possible to have fast response time because the data is located where it is accessed most frequently. Also, data replication allows the applications to be more available because applications can continue to run even if sites fail in the network.

Since there can be true parallel processing of multiple computers in the network, greater throughput can be achieved.

Greater Manageability

INGRES/Star's open architecture permits managers to maintain security and integrity of the data locally, while still having their data accessible across the network. Thus, the "owner" of each machine determines who may access a particular table, row, or column of data.[49]

Architecture

INGRES/Star offers an open-architecture distributed database. INGRES/Star really consists of two parts, namely, the distributed database manager and the local database manager. The distributed database manager takes care of managing the communications between the user applications and the local database manager. Whereas, the local database manager gets requests from the distributed database manager and takes care of processing the requests (see Figure 11). I have included Figure 12 in order to illustrate the fact that INGRES RDBMS accesses data differently than distributed INGRES. As you can see by comparing Figure 11 and Figure 12, the distributed database uses both a centralized distributed database manager to deal with queries and responses to the various local database managers. Under INGRES RDBMS, there is only one database, and hence only one local database manager is required.

Distributed Database Manager

The distributed database manager receives queries from the user application. Once it receives a query, the distributed database manager converts the query into subqueries based on the nodes where the information is stored. Second, it transmits these subqueries to the appropriate local database managers. The local database managers process the subqueries and then the distributed

Distributed Database Data Flow

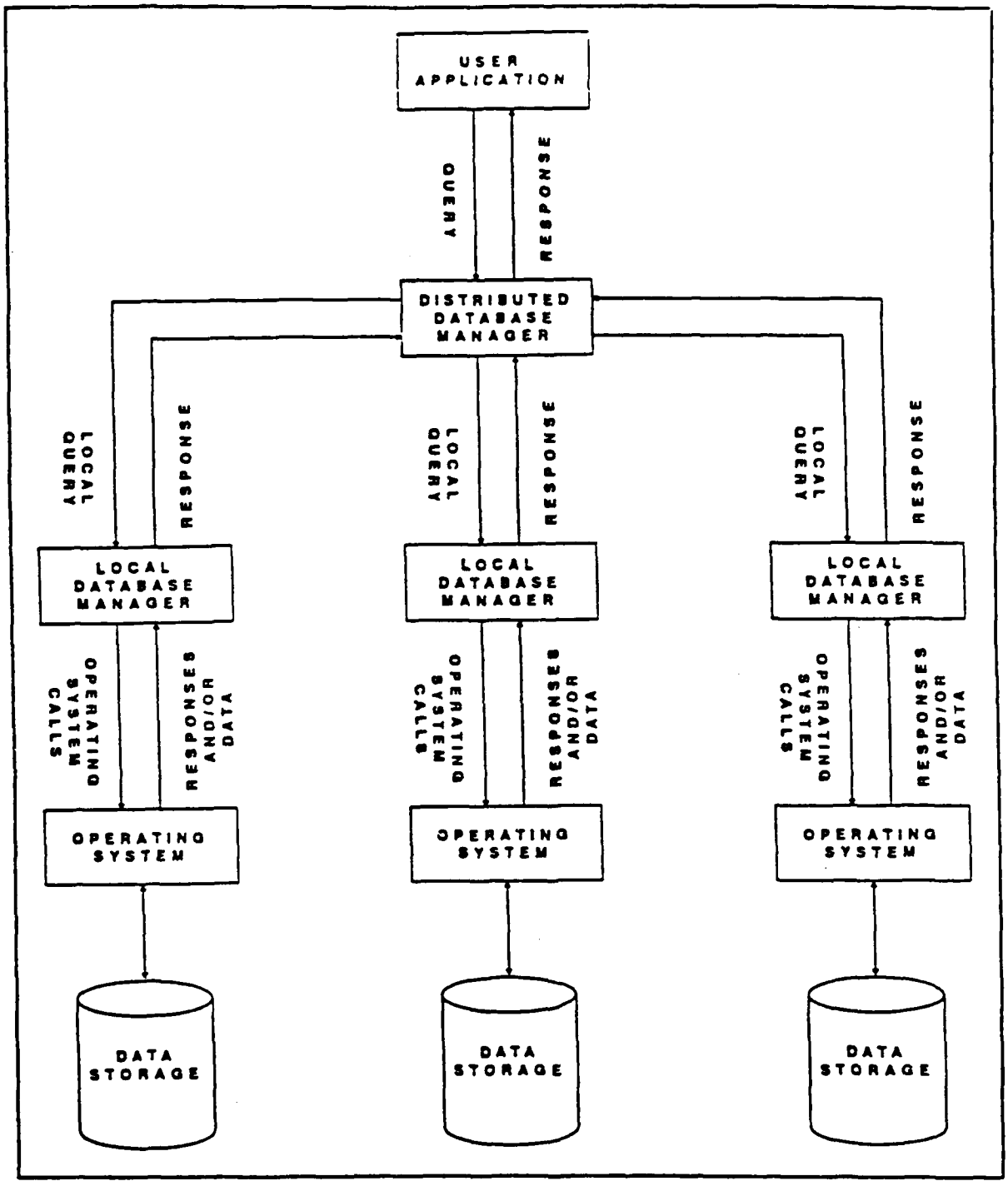


Figure 11

Local Database Data Flow

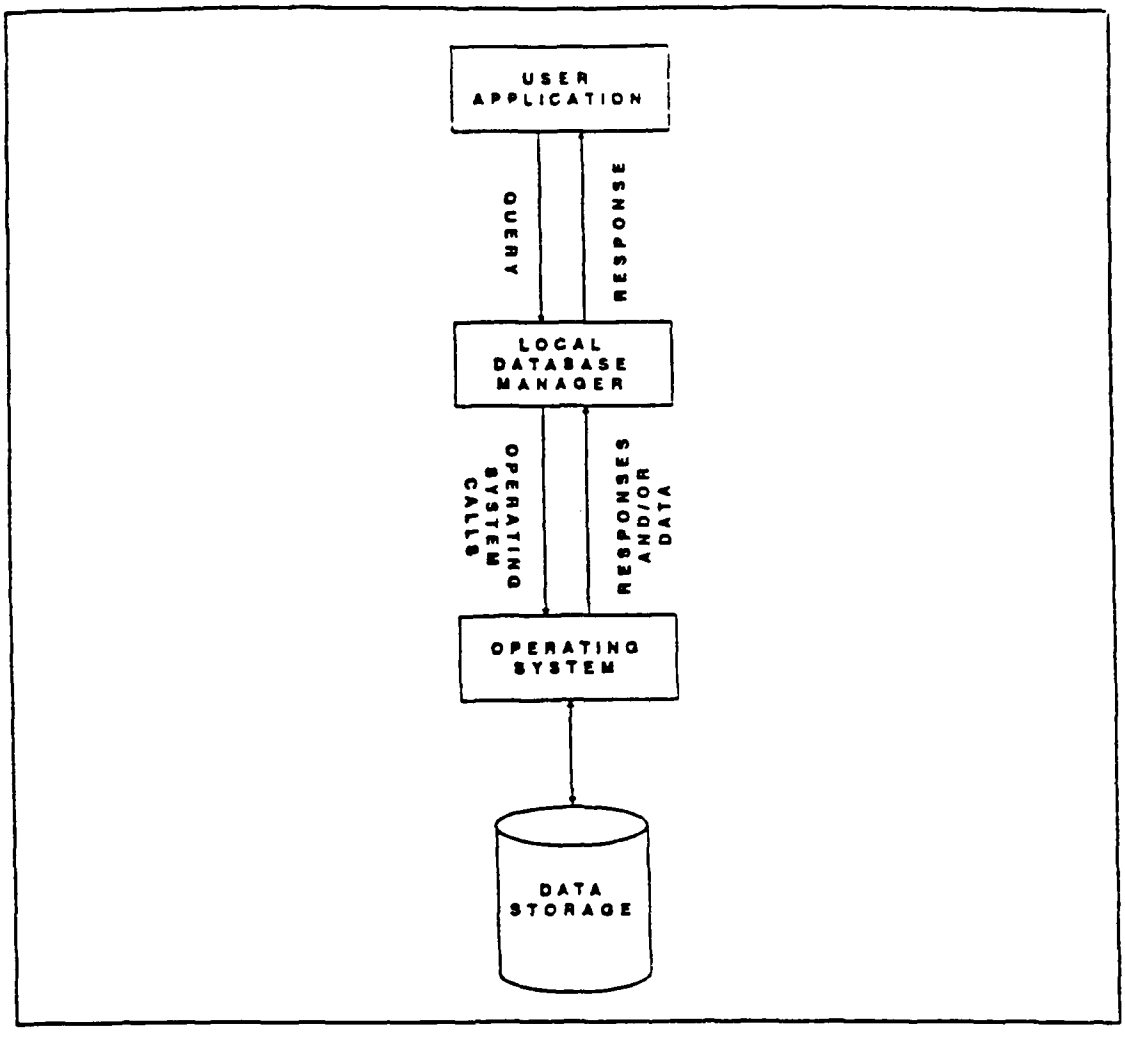


Figure 12

database manager gathers the results. Finally, the distributed database manager returns the results of the query to the INGRES application. Thus, INGRES's architecture is different from ORACLE because INGRES uses the centralized distributed database manager, whereas ORACLE does not.

Local Database Manager

Each computer in the network has its own local database manager. Thus, the local database manager is capable of handling simultaneous requests from the distributed database manager and users' requests for access to local databases.

Concurrency Control

INGRES LOCKING

INGRES uses page level locking rather than record-level locking since they feel it provides better overall performance for four reasons.

First, by utilizing a page level locking, you take advantage of reduced overhead since the operating system handles I/O based on 2K pages. Page level locking has less overhead than record level locking with regard to managing segments.

Second, it is easy to reorganize records on a page in order to avoid fragmentation (maximize use of disk space)

without additional overhead of locks.

Third, INGRES makes use of B-TREE and ISAM indexes at the page level. Because the locking is done at the page level, the indexes are fewer in number than under record level locking, thus faster sorting.

Fourth, INGRES provides a HASH access method. The HASH access method is good for exact key searches. For example, a social security number could be used to access a person's record immediately.

Fifth, in read only cases, the read level locks are shared. Thus, there is no increase in overhead. With record level locking there is more overhead because the locked unit is a record, whereas the I/O unit is a page.[50]

Query Optimizer Features

INGRES uses a statistics-based query optimizer. The INGRES query optimizer generates what is known as a Query Execution Plan (QEP). A Query Execution Plan is an optimization method used by INGRES which looks at the following factors when performing a query: amount of disk input/output, amount of CPU it will require, and the communications cost.

The primary advantage of the INGRES query optimizer is that it makes use of statistical histograms. Histograms are a pictorial view of the data and are generated based on the exact data in the database. Thus, the query optimizer takes advantage of this feature since it has knowledge of the data that will be needed to satisfy a given criteria. Furthermore, INGRES uses sophisticated mathematical modeling techniques to figure out how much CPU and I/O time will be required for a given query.

An added advantage of this query optimizer is that the QEP can be saved and reused if you are dealing with relatively static databases. Also, the QEP will draw pictures for you to show you exactly what is going on within a query.

Finally, both INGRES and ORACLE query optimization is completely transparent to the user. In other words, programmers and end-users do not have to design their own access methods to the data.[51]

Use of Personal Computers

INGRES on the personal computer requires: 640 K memory for application development and as little as 350 K for runtime applications, DOS 2.1 or later, and 5 Megabytes of

disk space. The INGRES Version 5.0 for the PC is a fully functional implementation of INGRES. INGRES for PCs allows the user to choose one of two possible interfaces. The user can use either the "classic" INGRES menu structure with menus at the bottom of each screen or he can choose to use a ring-style interface, similar to that found on Lotus 1-2-3. It is quite easy for a user to switch back and forth between the two different interfaces.[52]

According to a test performed by Palmer and Associates Inc., an independent consulting company, INGRES Version 5.0 for PCs outperformed ORACLE Version 4.0 for PCs in almost all categories. INGRES was exceptional in its performance (vs. ORACLE and Informix) with regard to the creation of large indices and importing and exporting data form external files.[53]

Compatibility

INGRES uses QUEL as its primary data access language which is interfaced with SQL.[54] INGRES was not fully compatible with IBM and ANSI SQL standards until very recently.

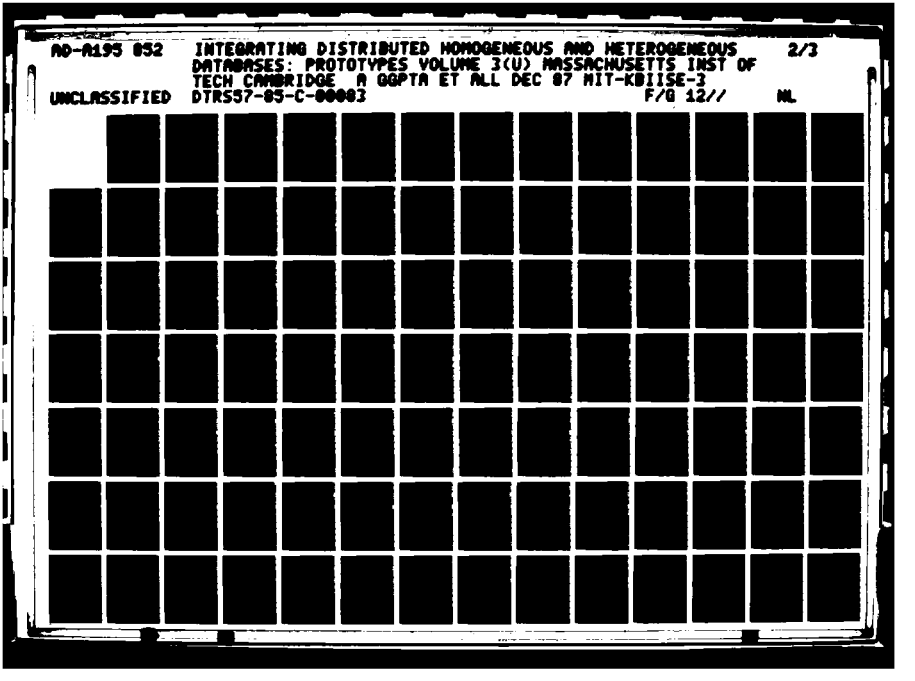
Portability

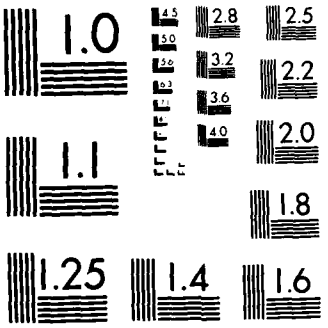
Currently, INGRES/Star runs on: IBM VM/CMS, DEC VAX/VMS, about twenty-four Unix machines and the IBM PC.

VMS System Interfaces

INGRES makes use of two VMS features. First, INGRES uses the native data types (ex. floating point, integer, character) offered by VMS. By using the VMS native data types rather than creating their own, it is possible to execute far less instruction sets. This is one instance where INGRES benefits by being less portable than ORACLE but more highly tuned. Second, INGRES uses the VMS distributed lock manager and in fact, builds it right into their product. This fact made it possible for INGRES to run on the VAX-Cluster as soon as it was introduced and with extremely high performance.

On the downside, INGRES was founded on the Unix operating system and does not take full advantage of the VMS operating system. For example, INGRES requires two tasks (and inter-process communication) for each active user probably because of the pipe structure in Unix. Thus, interprocess communication causes degradation of performance in this manner. INGRES does not utilize a shared buffer pool and hence causes an excessive input/output load.[55]


AD-A195 052 INTEGRATING DISTRIBUTED HOMOGENEOUS AND HETEROGENEOUS 2/3
DATABASES: PROTOTYPES VOLUME 3(U) MASSACHUSETTS INST OF
TECH CAMBRIDGE A GUPTA ET ALL DEC 87 MIT-KBIISE-3
UNCLASSIFIED DTRS57-85-C-00083 F/G 12// NL



Performance Related Issues

Advantages

INGRES supports a large variety of file access methods including both regular and compressed versions of: ISAM, HASH, HEAP, SORT-HEAP and B-TREE. The advantage of allowing this variety is that each method offers distinctive advantages depending on the situation. HEAP is good when you have sequential flat files and also for dumps. HASH allows extremely fast access and is good when you have a key number (example: Employee ID number). Indexed Sequential Access Method (ISAM) is a preferred method if you have ranges with relatively static data. The "all-purpose" access mode is B-TREE. Thus, the advantage INGRES offers here is that it gives you the ability to fine tune since you can choose an access mode suited to the situation at hand.

INGRES does have a method which allows you to copy records in batch rather than one at a time.[56]

Disadvantages

On the downside, INGRES does not support nulls. That is, INGRES does not differentiate between nulls, blanks and zero values. This might be important if you are using statistics or developing you databases incrementally. For

example, suppose we are developing a inventory database incrementally and would like to know about a particular part in inventory. There must be a distinction between "zero" (for no stock remaining) and "null" (we do not have any information on this item).

Also, INGRES limits tables to 127 columns and rows to 2,008 bytes and hence, does not support long text fields. This may be important if you have long textual fields that sometimes occurs in payrolls, phone books, etc.[57]

Query Language

INGRES originally used their own proprietary query language called QUEL which was only recently interfaced with SQL. INGRES now claims that their SQL is ANSI SQL compatible. Many INGRES users, including the large international bank in Chapter Seven, mentioned that they felt QUEL was more powerful and easier to use than SQL.[58]

4 GL Tools

Overall, most users seemed very happy with the INGRES tools saying they were fast and easy to use. However, some users felt the report writer was too slow and could be

improved.[59]

Security

INGRES does have many security features. Like ORACLE, it has a security audit feature so that violators can be tracked down. It is possible to query the data dictionary (if you have priority to do so) just like you would for any database. INGRES does seems to lack the sophisticated locking techniques mentioned above.

Network Support

INGRES only supports DECNET and TCP/IP. They do not provide support for any user customization to other networks as does ORACLE.[60]

CHAPTER FIVE

ORACLE VERSUS INGRES - A COMPARISON

What follows is a comparison between distributed INGRES and distributed ORACLE. What I have done in Chapter Five is to expound more fully on my evaluation column in my comparison table (see Figure 1).

Transparency

How ORACLE And INGRES Measure Up On Transparency

Since ORACLE and INGRES are constantly evolving systems, I will only evaluate what they can do at the present time (see Comparison Table - Figure 1). INGRES/Star obeys rules one through four and fails rules five and six, whereas ORACLE obeys rules one and two.[61] These differences result primarily due to the different architectures employed by INGRES and ORACLE. INGRES uses the central data dictionary approach, whereas ORACLE uses the site autonomy approach.

Independence

Crash Independence

When a crash occurs on a given node in SQL*Star, the only the users requesting data from the crashed node are affected. If a crash occurs in INGRES at the node which contains the distributed database manager then the entire system will be down. If a crash occurs in INGRES at any other node, then it would be the same as a crash experienced in ORACLE.

Recovery Independence

Recovery from a crash is automatic as can be seen in the R-Star research done at IBM.

Recovery under INGRES is much more difficult due to the architecture of the system, more specifically, the central data dictionary. Thus, if INGRES/Star crashes while executing a schema command, it would be necessary to have a coordinated recovery between the central data dictionary and the local system which crashed. Because INGRES does not support multi-system coordinated recovery, it is necessary to do a manual recovery in this situation.

Network Independence

As you can see in Figure 1, INGRES supports only TCP/IP and DECNET, whereas ORACLE supports a much larger set of protocols. In addition, ORACLE will support customers who

want to develop custom protocols.[62]

Hardware/OS Independence

ORACLE supports a much wider variety of hardware and operating systems than INGRES. INGRES supports VAX/VMS, Unix, and VM/CMS.

DBMS Independence

SQL*Star makes it easy to use non-ORACLE data from both DB2 and SQL/DS.

Concurrency

In ORACLE, readers do not block writers and writers do not block readers. In contrast, INGRES users who attempt to write a block must wait for all active readers to finish. Readers in INGRES cannot read a block which has been written already but not yet committed. Thus, the reader must wait until the writer is completely finished.

The implications of these differences are severe under applications with many users. Under INGRES, system and user performance will be degraded since much waiting occurs as each must wait until all others are finished before continuing. [63]

Deadlocks

There are two strategies for dealing with deadlocks. First, you could decide to prevent deadlocks before they occur. But this strategy requires much locking and releasing of locks which causes a degradation of performance. Second, you could allow deadlocks to happen since they occur infrequently, but have an automatic deadlock detection and recovery scheme. This is the method chosen by INGRES and ORACLE.[64]

Query Optimizer Features

I believe both INGRES and ORACLE have good query optimizers. INGRES uses a statistics based optimizer, whereas ORACLE uses artificial intelligence. Both optimizers allow efficient navigation among databases.

Use of Personal Computers

The INGRES PC version 5.0 is faster on almost all operations over ORACLE version 4.0. However, I believe that ORACLE's PC version is more fully functional in the sense that it acts like any other ORACLE system. INGRES has many things missing from its PC version compared with their VAX/VMS version. For example, only the B-TREE file access method is allowed on the INGRES PC version.

Compatibility

INGRES now claims to be fully ANSI SQL compatible. However, I do not think this is really the case because INGRES cannot support nulls. ORACLE has been fully ANSI SQL compatible since it began.

Portability

ORACLE is more portable than INGRES as you can see by examining Table 11. Although both INGRES and ORACLE are written in C, ORACLE ports their systems from the same source code. INGRES was originally founded on the Unix operating system, and thus it is more customized to this operating system. That is, ORACLE seems to be less customized to a particular operating system and thus can run a larger variety of operating systems and machines.

VMS System Interface

Both INGRES and ORACLE take advantage of certain features of VMS. ORACLE takes advantage of a shared buffer pool and shared code, whereas INGRES takes advantage of native data types and the VMS distributed lock manager.

IBM VM/CMS

ORACLE takes advantage of the shared global area (SGA) offered by IBM in order to maximize performance. INGRES uses the minidisk approach when dealing with IBM VM/CMS.

Performance Related Issues

I believe that ORACLE offers more performance related features than INGRES. ORACLE supports long text and has an array interface which allows you to bring over an array of records rather than just one at a time. ORACLE supports nulls, which may be very important if you an application where it is imperitive to differentiate between a zero value, and no value (null). ORACLE can also use multiple indexes to process a query whereas INGRES can use only one. INGRES does offer more file access methods. Thus, it is possible for the user to choose a file access method best suited to his application. For example, if you have a social security number you will probably want to use the HASH method since it would result in the best performance. The B-TREE method is the most all purpose method which both INGRES and ORACLE have.

Query Language

INGRES was orginally founded on the Unix operating system and used the QUEL query language. Recently, they claim they are fully ANSI SQL compatible. However, they do not support nulls and cannot handle the VAX packed decimal format. ORACLE, who pays close attention to what IBM is doing, made the strategic decision early on to be fully ANSI SQL compatible. Furthermore, ORACLE SQL has significant, powerful extensions to ANSI SQL. However, many INGRES users surveyed by DATAPRO, said that they liked QUEL and found it more powerful than SQL. However, SQL is the standard, although INGRES people say that they plan to continue to support QUEL.

4 GL Tools

Both INGRES and ORACLE seem to have good tools. However, the major government lab discussed in Chapter Eight, said that they would like to see more user-friendly interfaces using icons, pull down menus, etc.

Security

ORACLE seems to offer a very high level of security.

ORACLE provides a security audit facility so that it is possible to track down violators. Also, ORACLE provides view isolation and protected tables. In INGRES, it is possible to query the data dictionary (with proper priviledges) just as any database. Thus, it is possible to track down violators by querying the data dictionary which contains everything that is going on in the system.

Network Support

As you can see in Figure 1, ORACLE supports a much larger variety of networks than INGRES. Furthermore, ORACLE will support customers who want to develop custom protocols.

CHAPTER SIX

DISTRIBUTED DATABASE ISSSUES

In this chapter I attempt to examine some of the issues surrounding distributed database systems. More specifically, I will try to address what problems distributed database systems solve and what problems they create.

I believe one issue that comes up again and again is "Should an organization be centralized or decentralized with regard to information flow?" In most corporations, as the price of hardware dropped significantly they bought a variety of different machines. This obviously led to a decentralization of information - often each department had their own computer.

The primary advantage (i.e. problem they solve) of a distributed database system is that it allows you to tie the decentralized information back together. That is, without a distributed database system, you just have information floating around on different machines with no way to integrate it across the different hardware. Distributed database technology allows you to have one common view of the data without worrying about the particular hardware, or operating system.

Data Control

One problem that arises is "Who controls the information?". An individual user many say this is my data and I do not want anyone to touch my data. This is the basic approach taken by current distributed database systems because the security is imposed on a site by site basis. In other words, the owner of the machine is the one who controls who can and cannot access a particular table, column or row of data. This type of situation is seen as favorable by some people such as the personnel department, since they can control who can and cannot access salary information. However, from an administrative point of view it is much more difficult to control this matter so that people who should have the authority to see particular information can do so.

Optimization

An area which is likely to receive much attention in distributed database systems is optimization. Many additional situations arise in a distributed system (versus a traditional RDBMS) which require optimization. Consider the following two examples:

Example 1: There is a network of computers and there are seven different ways to get to a particular machine. You should choose the best route.

Example 2: There are a total of 6 machines. Four are busy and 2 are free. Thus, if you are doing a join, it might be better to use the free machines.

Neither ORACLE nor INGRES addresses the situations mentioned in Examples One and Two.

Another trend in the industry seems to be to develop gateways into other systems. By gateways, I mean one DBMS can transparently go into another system. In other words, the user will not have to be concerned with what DBMSs are involved when he performs a query. The reason this is such a "hot" area is because many dollars have been invested in other types of database management systems (example: IMS) and this old technology will not disappear within the next five or ten years. Thus, current distributed database systems should be able to work in conjunction with existing systems if they are to be truly successful.

Finally, let me speculate a bit on the future of distributed database systems. Once distributed systems become more developed they will be allow extremely high performance, that is, faster performance that can be attained on any single machine. Thus it will be possible to have multiple nodes working in parallel (i.e. parallel processing) on the same problem.

CHAPTER SEVEN

MINICASE STUDY ON A LARGE INTERNATIONAL BANK

Distributed INGRES Case Study

In 1982, a major international bank chose to use ORACLE's relational database management system. They chose ORACLE because at that time the type of production applications they were doing required high performance. More specifically, they needed to do many read operations very quickly and they found ORACLE appropriate under these circumstances.

However, over time, the bank found there was more and more demand to build applications at a faster rate, especially smaller applications. Thus, the bank found the fourth generation tools that went along with ORACLE not as effective, pleasant to use, or as fast as INGRES tools. Therefore, the bank chose to switch to the INGRES RDBMS primarily because they found the fourth generation tools of INGRES better than those of ORACLE.

Bill a consultant to the bank mentioned additional reasons why the bank chose INGRES which follow. Bill said INGRES' performance improved a great deal and that

INGRESappears to perform better at joins than ORACLE. He also said ORACLE seems to be better at single table access, however, he stated that INGRES has improved significantly in single table access. Furthermore, benchmarks they performed at the bank showed the systems were comparable in terms of their general performance.

Since the bank liked the INGRES RDBMS, they decided to stay with INGRES when deciding to use a distributed database system. In other words, when they found the need for a distributed database system they chose distributed INGRES because they liked the INGRES RDBMS they were using. Note that the bank did not do a comparison study between distributed INGRES and distributed ORACLE to arrive at their decision.

Was Security A Factor In Their Decision?

The bank said at the time they made their decision security was not an issue. They are using Digital Equipment Corporation's VMS operating system at the bank, so they claim that the VMS operating system provides many security features. However, security issues have become a concern recently.

Why Did They Need A Distributed Database System?

The first applications of distibuted INGRES are not really taking full advantage of the distributed system. The bank primarily has used a feature which INGRES has which allows you to run your application on one machine (front-end) with your database on another (back-end). It is possible to do this using ORACLE but it is a great deal of work. For example: In 1982 the bank used ORACLE RDBMS and put applications on multiple machines and the database on a single machine. Since a distributed database systems was not available at the time, they separated the front-end and the back-end. The bank used their own protocol for applications requesting data from the database and the database returned the data to the application (to the front-end).

In INGRES, the front-end is running on one machine (i.e. the terminal process) and the database is running on another machine. These two processes communicate with each other at the application level. The bank finds this feature extremely useful since they have many users around the world going against the same database and all these systems are being built on VAX machines.

Examples of Applications Using Distibuted INGRES

The distributed INGRES database system (more

specifically the 'front-end, back-end feature) was first used at the bank to work on various small applications (around 30 simultaneous users at each site) used in many sites around the world. If the bank allowed the entire application to reside at each site, they would also have to provide full backup facilities at every site. Thus, by putting the front-end out at each site and centralizing the databases in New York City all the backup is done in New York City and hence all the redundancy is in NYC. So, in fact if one machine goes down then the users directly access the application in New York City.

Second, they are building a global system right now in order to resolve the following problem: They have transactions for a customer of site A which are executed by a customer at site B. Without a distributed database system, the various position and transaction records are duplicated at the application level at both site A and site B. Hence, you run into problems when the data is different at sites A and B and therefore have to reconcile the records. Thus, in this global system it is necessary to reconcile the records around the world.

With a distributed database system, if they do want to replicate the data it is purely a performance issue done within the database system and not at the application level. So from a logical point of view there is a single record at

each site. Therefore, there is no reconciliation problem. Apparently, this reconciliation problem currently exists for many applications at the bank, such as:

- Securities custody: This system involves a settlement of security transactions around the world.
- Foreign exchange
- Funds transfers

In order to give an idea of the typical transaction volume found on one of these systems, let us look at the securities custody application. The current volume on the securities custody system is approximately 10,000 transactions per day with about 5,000 of these transactions performed in New York City.

Are You Satisfied With Distributed INGRES?

The bank says that INGRES is still in the Beta Version and that they have only really used the front-end, back-end features associated with the distributed database systems.

Bill says distributed INGRES does not have the following two features which the bank needs which are two phase commit, and deferred copies. However, the bank expects INGRES to have these features incorporated into their product by the end of 1987.

Other Comments On INGRES

Until recently, INGRES has seemed to take the lion's share of the scientific and academic market whereas ORACLE has the lion's share of the business market.

By supporting SQL, INGRES has managed to make inroads into the business market. The bank said that INGRES' SQL was not so good when the bank started using it one year ago. Now, however, the bank said that INGRES' SQL is more fully integrated into Relational Technology's various products. INGRES now claims to be fully ANSI SQL compatible.

The bank used SQL for three years and then they started using QUEL and they found it better. Their comment was that QUEL seems more straightforward and flexible. Bill feels that QUEL makes it easier for the user to say what they want to say and he feels QUEL is more powerful than ANSI SQL. However, because SQL is a standard, they feel QUEL will not be supported by early 1990.[65]

CHAPTER EIGHT

MINICASE STUDY ON A MAJOR GOVERNMENT LABORATORY

Distributed ORACLE Case Study

Distributed ORACLE was chosen by the major government lab because an RPF was put out to anyone who wanted to bid and ORACLE responded, whereas INGRES did not. Currently, the lab is running distributed ORACLE under VMS Version 4.5. AS far which system is better, John Rector of the lab said both INGRES and ORACLE have their strong points (these points are mentioned later in this minicase).

Why Did The Lab Need A Distributed Database?

When the lab bought ORACLE, it was with the intent that it would be distributed. The only reason they had used non-distributed ORACLE was because distributed ORACLE simply had not been developed yet. The lab said they were fully aware when they purchased non-distributed ORACLE of the time frame involved to receive the distributed version.

How The Lab is Using Distributed ORACLE

The lab says essentially they are a local area network (LAN). For instance, someone may want to do a large project and he may need data that is collectively owned. Then he may want to run something which is CPU intensive. For example, a new tool might use much CPU power because it has a screen manager. Therefore, it is nice to put that user off on a Micro-VAX and then have him be able to select his data from a larger server (for example, an 8550). Thus, the users actual computational power comes from some smaller machine whether it be a personal computer or a VAX station. Therefore, the idea is to make use of database servers as well as to give individuals more computing power but on a separate CPU.

John Rector said accessing the same data really has two components. First, there is now one place to get at the data. Second, the data, which is a very important commodity, can be controlled by someone who is trained in maintaining it. For example, you can have a database administrator (DBA) or a systems manager control the 8550's and databases. This makes it possible for users to pull the data over and do what they want to it. It is not feasible to have a DBA looking at each one of the individual databases because the costs of doing so are too expensive.

<u>How Much Degradation Is There in LAN?</u>

The following statements apply to both ORACLE and INGRES.

First, let us assume that a user brings up a background process (fires up a server). This user has got some image running as a process on a local computer but he is also firing up an image in another process on the host computer. So there are a couple of things which are happening here. That is, whenever you do a connect to a database, you incur the full overhead (operating system overhead) of creating a process. This overhead is certainly significant. Now, if there are many people making this type of request on a small VAX, for example 20 or 30, that is a significant amount of time that the operating system is spent simply generating new processes. Even just context switching between them may be significant. You can on a VAX (any OS, though) see how much time it is spending doing operating system things. In VAX terminology, you can turn on the monitor and see how much time the OS is spending in kernel mode and how much time it is spending in user mode. This will give you an idea of the impact of the database as far as how much time is spent serving the user and how much time it is doing all the overhead. Thus, overhead increases substantially with distributed database systems.

Also, another thing that enters the picture is the

design of programs and the design of methodologies of how user's access these things. If users are continually logging in to the database, doing something, thenlogging off, the load increases tremendously as opposed to if they logon and stay connected. One thing you can do if you have a workstation is you can switch between processors and keep that connection open all the time. So from a user point of view that makes the response time quicker because the user does not incur process creation overhead.

High Performance Feature of ORACLE

John Rector mentioned ORACLE's feature called an Array Fetch which lets you fetch multiple rows of data into the buffered space in your program. He said this is a significant performance feature because it allows you to pull data over in an array as opposed to a row at a time.

Is Security An Issue?

Security is a very important issue at the lab. John Rector said security is handled very well by ORACLE. He said ORACLE followed the System R design with regard to security and thus provides the GRANT command. Also, ORACLE added the AUDIT command which allows you to track down system

violators.

General Security Problems Related to Distributed DBMS

One of the things not fully addressed in a distributed environment is access to the communication lines, but that is not necessarily a DBMS problem per say, but it is still a problem which must be addressed. John Rector says you now encounter all the security problems you have whenever there are communication lines coming into your system of any sort. This network problem becomes a database problem because you are using that facility. Thus, you are open to a few more serious security problems when using a distributed DBMS.

How Do You Find ORACLE's Tools

John Rector said that ORACLE's ad hoc processor, SQL*Plus is a good tool.

However, John thinks neither INGRES nor ORACLE is leading in the 4 GL tool area. He said you do not see ad hoc processors of the type you currently see in the workstation, MacIntosh, or even the new IBM PC environment. That is, you do not see user-friendly tools which use a mouse, pull-down menus, etc. Both companies' marketing approach has suggested

that everyone is using a VT 200 terminal rather than a workstation.

John Rector said the ORACLE Forms Product is approaching the workstation or MacIntosh mentality with the design of a form done with pull down menus. Right now on the VT 220 it is necessary to use arrow keys, however, a mouse interface is planned.

On the other hand, John said ORACLE's Report Generator product is a very archaic product. For example, when you build a file you use .S in order to put a space in. John said that there are third parties who provide a Report Manager. He also said you will see more and more involvement of third parties since SQL is the standard. In other words, it is now cost justifiable for third parties to enter the market.

Other Improvements Needed For Distributed Database Systems

First, John says that the distributed database companies need user feedback since they are brand new. Furthermore, they need time for the design of peripherals that go with distributed systems.

Second, John feels that database companies should get

away from the "single-tube" concept and move toward tools which are based more on the workstation style of user interface (example: MacIntosh).[66]

CHAPTER NINE

CONCLUSION

Managing A Distributed Database System

Neither ORACLE nor INGRES has really addressed the issue of how to manage a distributed database system.

INGRES uses one solution, namely a central data dictionary. The problem with this is if the node goes down containing the distributed database manager, then you have serious problems.

ORACLE uses the concept of adding a user to a node. Obviously, you can add a single user to many nodes. Thus, this architecture seems designed for long haul databases (example: NY, LA, Chicago). The idea is that people want to communicate between the sites but essentially the sites are separate entities. In other words, there is some communication between the sites, but not a great deal.

A drawback of adding node by node is that you need a little bit more technical expertise in order to add node by node rather than by a central dictionary approach.

There is a political issue when dealing with distributed

database systems as well. For example, who owns a particular VAX machine becomes a political issue. Many things do not happen in databases not due to technical reasons but rather, because of the politics of the organization and who owns what. This is a problem because how do you manage users on a distributed system or how do you maintain a distributed system (how do you do backups, etc)?

A concept you are starting to see is LAN with, for example, four or five VAX workstations and three 80350s. You may want to add a user to all 80350s, that is you simply want to grant him access (select) on some table. That table may even move from one site to another. This issue has not been well addressed, although it may just be early. Furthermore, some of the DBA tools have not been expanded to the point where they allow a DBA to gracefully handle a distributed database.[67]

Future Directions

Both distributed ORACLE and distributed INGRES offer the user a great deal. That is, they allow the user to access data transparently from a variety of machines and operating systems.

However, both ORACLE and INGRES must add other features

such as: (1) full update capabilities, (2) support concurrent copies, and (3) provide gateways to non-SQL DBMSs.

Full Update Capabilities

This feature will allow users to update multiple sites in a single transaction. Thus, it will be possible to issue a single update command which will affect several databases.

Support Concurrent Copies

This feature could make it possible to have secondary copies of tables which will be updated concurrently and transparently to the user. The concurrent copies feature makes it possible to keep the data close to the users and thus serves as a performance feature (as well as minimizing communication costs).

Gateways Into Non-SQL Systems

Both ORACLE and INGRES need to work on building gateways into other systems. For example, many companies have invested a great deal of money in IMS applications. It would be nice if ORACLE and INGRES could access this data so that the user would not even realize he is dealing with IMS.

FOOTNOTES

(1) "INGRES/Star Distributed Relational DBMS Technical Backgrounder", May 21, 1986.

(2) "INGRES/Star: The Distributed SQL Relational Database"

(3) "INGRES/STAR"

(4) Neville, Donna. ORACLE: SQL*Net User's Guide. Belmont, CA: Oracle Corporation. August 26, 1986, p 2.

(5) INGRES/STAR Administrator's Guide - Release 5.0 VMS Version. Alameda, CA: Relational Technology Inc.. 1987, Chapter 2, p 1.

(6) "Relational Technology Announces Breakthrough With Distributed Database Product". June 9, 1986, p 5.

(7) "Query Processing in R*", p 31.

(8) Ibid. p 32.

(9) Stonebraker, Michael. "Transparency in Distributed Database Systems". p 1.

(10) Ibid. p 2.

(11) Ibid. p 3.

(12) Ibid. p 4.

(13) Memo from Ken Cohen of Oracle Corporation, p 5.

(14) Ibid p 6.

(15) Ibid, p 7.

(16) SQL: The Quiet Revolution. ORACLE. ORCE Systems Software B.V. 1986. p 106.

(17) SQL: The Quiet Revolution. ORACLE. ORCE Systems Software B.V. 1986. p 106.

(18) DATAPRO RESEARCH CORPORATION, "Oracle Corporation: ORACLE". Delran, NJ. November 1986, p 101.

(19) Shamoon, Sherrie. Management Technology. "Oracle's Larry Ellison: The king of relational software". December 1984. Reprint.

(20) DATAPRO RESEARCH CORPORATION, "Oracle Corporation:

ORACLE". Delran, NJ. November 1986, p 102.

(21) Letter from David Martin of Oracle Corporation

(22) DATAPRO RESEARCH CORPORATION, "Oracle Corporation: ORACLE". Delran, NJ. November 1986, p 101.

(23) Rizzo, Tony. PC Magazine. "Project Database II: Programmable Relational Database - ORACLE". June 24, 1986. p 146-147.

(24) DATAPRO RESEARCH CORPORATION, "Oracle Corporation: ORACLE". Delran, NJ. November 1986, p 103.

(25) Ibid. p 104.

(26) "Oracle - Products and Services Overview"

(27) Neville, Donna. ORACLE: SQL*Net User's Guide. Belmont, CA: Oracle Corporation. August 26, 1986, p 2.

(28) Ibid. p 9.

(29) Ibid. p 10.

(30) SQL: The Quiet Revolution. ORACLE. ORCE Systems Software B.V. 1986. pp 107-108.

(31) DATAPRO RESEARCH CORPORATION, "Oracle Corporation: ORACLE". Delran, NJ. November 1986, p 107.

(32) Rizzo, Tony. PC Magazine. "Project Database II: Programmable Relational Database - ORACLE". June 24, 1986. p 146.

(33) Ibid. p 147.

(34) Interview with John Callandrello of Relational Technology Inc.

(35) DATAPRO RESEARCH CORPORATION, "Oracle Corporation: ORACLE". Delran, NJ. November 1986, p 106.

(36) DATAPRO RESEARCH CORPORATION, "Oracle Corporation: ORACLE". Delran, NJ. November 1986, p 105.

(37) DATAPRO RESEARCH CORPORATION, "Oracle Corporation: ORACLE". Delran, NJ. November 1986, p 107.

(38) Neville, Donna. ORACLE: SQL*Net User's Guide. Belmont, CA: Oracle Corporation. August 26, 1986, pp 7-8.

(39) Stonebraker, Michael. The INGRES Papers: Anatomy of a Relational Database System. Reading, MA: Addison-Wesley

Publishing Company. 1986. p iii (Introduction).

(40) Ibid. p 64.

(41) DATAPRO RESEARCH CORPORATION, "Relational Technology Inc.: INGRES". Delran, NJ. December 1986, p 102.

(42) Ibid. p 102.

(43) Ibid. p 101.

(44) "INGRES/STAR". p 3.

(45) "INGRES - The Distributed SQL Relational Database System"

(46) "INGRES/Star Distributed Relational DBMS Technical Backgrounder", May 21, 1986. p 5.

(47) INGRES/STAR Administrator's Guide - Release 5.0 VMS Version. Alameda, CA: Relational Technology Inc.. 1987, Chapter 2, p 1.

(48) "INGRES/Star Distributed Relational DBMS Technical Backgrounder", May 21, 1986. p 5.

(49) Ibid. p 6.

(50) Interview with John Callandrello and Jim Milbery of Relational Technology Inc.

(51) The INGRES Advantage. "The INGRES Expert Query Optimizer". Fall 1986. pp 5-6.

(52) The INGRES Advantage. "INGRES Available for PCs". Volume II, Number 1. 1987. p 4.

(53) Margolis, Nell. Digital Review. "PC Ingres Blows by Oracle V4 in Beta-Test-Site Benchmark". p 1.

(54) Babcock, Charles. Computerworld. "DBMS Contenders In Grudge Match". March 16, 1987. p 121.

(55) Memo from International Network of Food Data Systems

(56) Interview with John Callandrello of Relational Technology Inc.

(57) DATAPRO RESEARCH CORPORATION, "Relational Technology Inc.: INGRES". Delran, NJ. December 1986, p 104.

(58) Babcock, Charles. Computerworld. "DBMS Contenders In Grudge Match". March 16, 1987. p 121.

(59) DATAPRO RESEARCH CORPORATION, "Relational Technology Inc.: INGRES". Delran, NJ. December 1986, p 105.

(60) Interview with John Callandrello and Jim Milbery of Relational Technology Inc.

(61) Stonebraker, Michael. "Transparency in Distributed Database Systems". p 5.

(62) Oracle memo by Ken Cohen

(63) Interview with Ken Jacobs of Oracle Corporation.

(64) SQL: The Quiet Revolution. ORACLE. ORCE Systems Software B.V. 1986. pp 107-108.

(65) Interview with Bill of a large international bank.

(66) Interview with John Rector of a large government research laboratory.

(67) Ibid.

BIBLIOGRAPHY

Babcock, Charles. Computerworld. "DBMS Contenders In Grudge Match". March 16, 1987.

DATAPRO RESEARCH CORPORATION, "Oracle Corporation: ORACLE". Delran, NJ. November 1986.

DATAPRO RESEARCH CORPORATION, "Relational Technology Inc.: INGRES". Delran, NJ. December 1986.

"INGRES/STAR"

INGRES/STAR Administrator's Guide - Release 5.0 VMS Version. Alameda, CA: Relational Technology Inc.. 1987.

"INGRES/Star Distributed Relational DBMS Technical Backgrounder", May 21, 1986.

"INGRES/Star: The Distributed SQL Relational Database"

"INGRES - The Distributed SQL Relational Database System"

Interview with Bill of a large international bank.

Interview with Ken Jacobs of Oracle Corporation.

Interview with John Callandrello and Jim Milbery of Relational Technology Inc.

Interview with John Rector of a large government research laboratory.

Margolis, Nell. Digital Review. "PC Ingres Blows by Oracle V4 in Beta-Test-Site Benchmark".

Memo from International Network of Food Data Systems

Neville, Donna. ORACLE: SQL*Net User's Guide. Belmont, CA: Oracle Corporation. August 26, 1986.

"Oracle - Products and Services Overview"

Oracle memo by Ken Cohen

"Query Processing in R*".

"Relational Technology Announces Breakthrough With Distributed Database Product". June 9, 1986.

Rizzo, Tony. PC Magazine. "Project Database II: Programmable Relational Database - ORACLE". June 24, 1986.

Shamoon, Sherrie. Management Technology. "Oracle's Larry Ellison: The king of relational software". December 1984. Reprint.

SQL: The Quiet Revolution. ORACLE. ORCE Systems Software B.V. 1986.

Stonebraker, Michael. The INGRES Papers: Anatomy of a Relational Database System. Reading, MA: Addison-Wesley Publishing Company. 1986.

Stonebraker, Michael. "Transparency in Distributed Database Systems".

The INGRES Advantage. "The INGRES Expert Query Optimizer". Fall 1986.

The INGRES Advantage. "INGRES Available for PCs". Volume II, Number 1. 1987.

ACHIEVING A SINGLE MODEL FOR INTEGRATING HETEROGENEOUS DATABASES

DANIEL KENNEDY

By virtue of the fact that heterogeneous database management systems must interconnect multiple databases together, a model for this area must necessarily address the database issue as well as the communication issue. This in turn motivates a study of both these issues, to come up with a single unified model.

In the communication area, the most common model is the one developed by the International Standards Organization for Open Systems Interconnection. Instead of choosing one framework, this model unfortunately standardizes two different kinds of network services: connection-oriented and connectionless. This duality was caused by the debate over datagram service versus virtual circuit service. The connectionless network service is based on experience with experimental networks, such as Arpanet, in which each packet of data travels independently carrying with it the information necessary to enable gateways to forward the message correctly. The connection-oriented network is similar to a telephone network and a connection is established before data can be transmitted to a particular destination. The undesirable existence of two dissimilar network services is reflected in the reference model.

In the database area, the commonly used model has been developed by the ANSI/X3/SPARC Study Group. Their primary focus was on local databases resident on one machine. An application is viewed in terms of three schemas - an external schema, a conceptual schema, and an internal schema. The ideas of the ANSI Study Group have been refined by the group currently involved in developing PDES (Product Definition Exchange Standard).

A single model, covering both areas, should consist of modified versions of the existing models. Specially, the ISO model should be modified so that the network level should be designed using a connectionless network protocol. Also, the application layer needs to be expanded.

TECHNICAL REPORT #16

Chapter 1

Heterogeneous Computer Networks

A recent trend in computer systems research has been towards the integration of and experimentation with heterogeneous computer networks. Passing information among processors with different internal data formats has proven to be a major complication to these computer networking efforts. At the same time, exchanging information between databases on machines with different architectures has also become a very difficult problem.

1.1 Networking

A computer network is a collection of computers, called "hosts" that can communicate with one another. A host can be anything from a personal workstation to a large supercomputer. "Dumb" terminals are not considered hosts. One definition says

> ...a computer network is defined to be a set of autonomous, independent computer systems, interconnected so as to permit interactive resource sharing between any pair of systems. [<Roberts> p. 543.]

There are two types of networks. They are local and long haul networks. A local area network (LAN) is used to connect computers in the same or adjacent buildings. The cables and interfaces used in an LAN achieve high speeds by taking advantage of lov error rates possible over short distances. A long-haul network is used to connect computers over long distances. Its interfaces and connections are primarily telephone or satellite links. The transmission rates are significantly lower for long-haul networks. Long-haul networks are typically operated by outside organizations. Local area networks are normally operated by the same organization that owns the computers.

1.2 Heterogeneous Machines

In most networks it is typically the case that the hosts are built with dissimilar architectures. There are two reasons for this in LAN's. The first is that when newer and faster computers were purchased, they were not purchased to replace the existing computers, but rather to supplement them. This reason is primarily economic. It would not be economical to replace all of the machines in a company every two years when the new line comes out. The capital investment represented by these computers, in both hardware and software, often prohibits their replacement with more compatible counterparts. The second reason is that computers from different vendors may be required to fulfill different purposes. For example, a company may have a large database of customers: in which case it would require a computer with large memory to store the database. However, a faster machine may be required to run applications that use the database. This would require connecting the two machines together. This would require some kind of heterogeneous computer network. It should be obvious that long-haul networks are primarily heterogeneous because they connect machines from several different organizations. The task is to accurately exchange data from one machine to the next with the user having minimal intervention into the exchange.

1.3 Homogeneous Machines

One way to solve the problem of heterogeneous computer networks is to avoid it. This can be accomplished by interconnecting machines that use similar internal data representations. Processors in such a homogeneous computing environment require no data format translation to exchange information. They are assured by common hardware and software design that the semantic content of their passed

data will be correctly understood by their intended receiver if the information is delivered correctly.

Obviously, homogeneous machines provide a processing environment more hospitable for inter-computer message transfer. Unfortunately, many organizations already have heterogeneous machines, as described above. Ignoring the data translation problem because it can be avoided in homogeneous environments is being unresponsive to the real needs of a large segment of the computing community.

1.4 Incompatibilities of Heterogeneous Machines

Conveying meaning of transmitted bits in a heterogeneous environment is not simple. The difficulty arises because of the lack of an industry standard for the internal representation of information in computers. There are machines of every description: they support sign-magnitude, or one's or two's complement arithmetic, 12, 16, 24, 32, 36, 48, or 60 bit word lengths, and unique floating point number representations. At the software level, there are different ways to represent complex numbers, vectors, arrays, and other data structures. There are even discrepancies in the case of character data. Although the ASCII character set is the industry standard, different machines still have different meanings for control characters such as form feed, line feed, tab, and carriage return.

In order to achieve a compromise between all of these technical differences a model is needed to correlate design principles, such that computers with dissimilar architectures can share databases. Two such models currently exist. They are the International Standards Organization Open Systems Interconnections model and the American National Standards Institute three-tier database model. These

models do not intend to recommend a particular protocol for transferring data. Instead, they model a system from a level above the particular implementation chosen for a protocol. In this way computers with different protocol implementations will be able to connect as long as the protocols can be adapted to have the same meanings. The models will be discussed in the next chapter.

Chapter 2

Existing Models for Integrating Computer Networks

Despite a wide variety of implementations of heterogeneous networks and a massive amount of technical detail, two models have emerged for representing such systems. Both models attempt to capture the problems involved in integrating disparate systems. While they may seem broad and general in nature, this is understandable. It would be impossible to develop a model that is specific enough to encapsulate all of the necessary information to connect every type of computer to every other type of computer. Therefore, the description of a model of such a system must be very general. Only model those components that are essential to every computer system can be modeled.

2.1 The International Standards Organization Model for Open Systems Interconnection

The first model that tried to capture these principles was the Open Systems Interconnections (OSI) model. It was developed by the International Standards Organization (ISO) in 1980 [<Tanenbaum2>]. The model organizes the functions of a network into a hierarchy according to their characteristic time scales and levels of abstraction. Each layer builds on, and adds functions to the layer below. The most fundamental graphic representation of the ISO reference model is seen in figure 2-1.

This is a copy of the reference model provided in ISO/TC97/SC16/N719. The diagram shows two seven layer structures resting on a physical media base. The

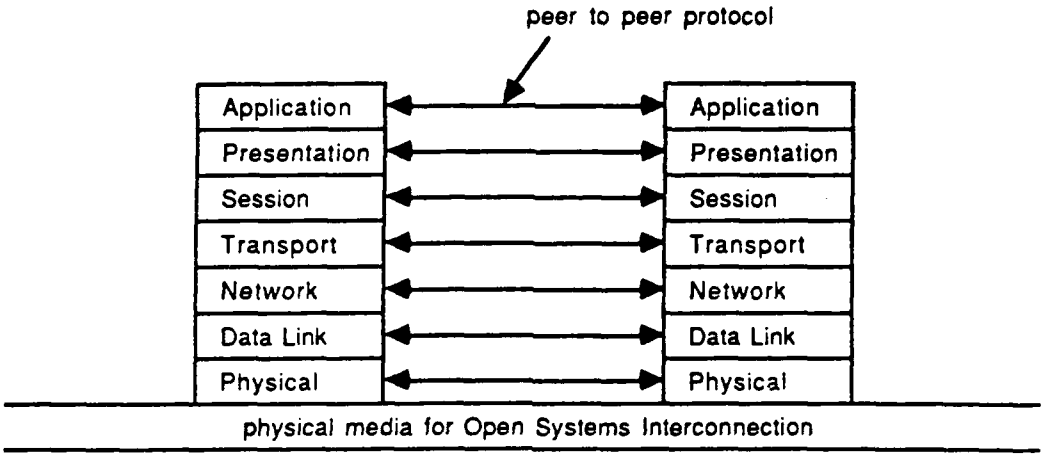


Figure 2-1: Seven layer ISO reference model

boundaries between layers define the point at which a layer can request the services of the layer below. Communications between layers on the two different structures are by means of protocols. Protocols are illustrated by the dashed lines between structures and are limited to communications between the same layers on both structures. "The ISO Reference Model of Open Systems Interconnection provides the functionality for interprocess communication between application processes." [<Bachman> p. 36]

The software and hardware implementing these layers must be present in each host in the network. The protocols in the figure show how the software is designed so that layer *i* on one host can interact with layer *i* on another host as if the lower layers did not exist. This is implemented by passing data from layer *i* on one host through the software on layers *i*-1,....1, being transformed by each as they pass through. This process is then performed in reverse-order on the other host until the data reappears on Layer *i*.

Here is a brief overview of each of the seven layers:

2.1.1 The Physical Layer

This is the hardware level at which the actual exchange of raw bits takes place. All electrical and mechanical aspects of data communication are handled at this level. Important design issues to be considered are how to represent bits as signals, whether to use half or full duplex communications, what pin configurations the connectors will have, and what type of network the host will be connecting to.

The user of layer 1 can be sure that a given string of bits will be encoded and transmitted. However, the user cannot be sure that the data has passed successfully over the data link. Errors are detected and corrected by layer 2.

2.1.2 The Data Link Layer

This level provides reliable physical links between adjacent hosts. In general, this is done by dividing the data into chunks, called frames, and then embed each frame into a packet for transmission. A packet also contains additional information such as destination address, a sequence number, and a checksum. These are used for detecting transmission errors. When the receiving host receives a packet it sends an acknowledgment back to the sending host. The sender will *re-send* a packet if it hasn't received an acknowledgment after a certain amount of time.

Data link protocols must also contain a mechanism that is known as flow control. This mechanism detects when the receiving host cannot receive packets as fast as the sender is sending them, and reacts by temporarily shutting off the sender. It is able to detect this by counting the number of unacknowledged packets.

The user of the data link layer is guaranteed that a given string of bits will be transmitted properly over a given link. The data link layer cannot, however,

send information over multiple link paths. Layer 3 sets up and manages multiple linked paths.

2.1.3 The Network Layer

This level provides multilink paths from host to host. The basic strategy is to place a "routing table" in each switch that tells what link to use when forwarding information to a given host. This routing table can either remain the same for an entire call, or it can be dynamically updated, depending on which network protocol is used. The most common network protocols are virtual circuits and datagrams.

Using virtual circuits means that every packet for a given message is passed along the same path. This path is established at the time when the message enters the network. The advantage is that once the path is established the exchange of data is very efficient. The disadvantage is that a lot of memory can be wasted remembering paths over which there is little traffic. Another problem is that if a node goes down a whole new path must be established. Also, while it is true that transmission errors will be caught, they will not be caught until they reach their ultimate destination. So if an error occurs at the first node in the path, it is not noticed until the packet reaches the receiving host. This is inefficient.

A datagram is a packet that is sent independently of all other packets from the same message. It must contain the full address of the receiving host. The advantage of datagrams is that messages that are small enough to fit into a single packet are sent very efficiently. This is because each time a packet reaches a new node it is sent off to the "best" node in the direction that it is going. Another advantage is that loads on links become more balanced. Also, better error detection and correction is achieved with datagram service. The disadvantage is that more packets are likely to get lost and packets can arrive out of sequence and thus must be re-organized. Layer 4 insures the reliability of these multilink paths.

2.1.4 The Transport Layer

This layer provides reliable multilink paths between pairs of hosts. The software includes tests to verify that circuits remain open or datagrams are eventually acknowledged.

The user of the transport layer can be sure that messages will be reliably delivered to remote hosts regardless of the state of the network, number of links on the path, datagram or virtual circuit service, or number of operational switches. At this level, the details of the network technology are completely hidden. The same interface can be used with networks ranging in speed from telephone lines to satellite links. Interprocess connections are managed by layer 5.

2.1.5 The Session Layer

This layer establishes and manages reliable connections between pairs of processes on different hosts. It is a small extension of the transport layer. It performs such functions as allowing symbolic names to be used when opening and closing connections, or matching responses from remote processes with multiple outstanding requests to those processes. The fifth layer deals with the transformation of data.

2.1.6 The Presentation Layer

This layer transforms data in certain ways as it moves between user programs and the network. It deals with such functions as:

1. encrypting messages. The contents of messages are encrypted on transmission and decrypted on receipt.

2. text compression. It is inefficient to transmit redundant data. There is usually a lot of redundancy in text data. Text compression can drastically reduce the amount of data to be transmitted.

3. virtual terminal protocol. This allows users to write programs that will work on any terminal in the network. This is accomplished by translating between commands on the actual terminal and a defined, "universal" set of virtual terminal commands.

2.1.7 The Applications Layer

This final layer is where all user programs that interact with the network must reside. This would include electronic mail programs, file transfer programs, and any database management systems that require remote access to data that exists on other machines in the network.

2.1.8 Relevance to Heterogeneous Databases

While the OSI reference model deals primarily with interconnecting machines and not databases, it should be noted that it is extremely relevant to the problem of interconnecting heterogeneous databases. When we talk about databases we are primarily dealing with the presentation layer and the application layer. However, it is of vital importance that the entire model is understood. This will be discussed further in Chapter 3.

The OSI reference model deals primarily with the transfer of data over a network from the perspective of the system. It does not model this transfer from a programmer perspective. This probably resides in the presentation and application layers. It has no formal description in this model, however. The next model to be looked at is the ANSI/X3/SPARC Study Group's framework for database management systems. It is more focused on the programmer's perspective.

2.2 The ANSI Framework for Database Management Systems

The purpose of the ANSI/X3/SPARC Study Group on Database Management Systems is to investigate the development of standards for database management systems. This group has developed a framework for description of such systems.

The purpose of this framework is not the same as the purpose of the ISO OSI reference model. As would be expected, the purpose of the study group's model is to model databases rather than network topologies. This means that it must not only look at data from the system's perspective, but also from the programmer's perspective. The ISO model uses the system's point of view. This is not a criticism, but rather a comment. The original purpose of the ISO model was to model networks from the system level. It satisfied its purpose. A more detailed discussion of this will be carried on in Chapter 3.

2.2.1 Description of the framework

In attempting to solve the problem the study group decided that the prime concern in modeling databases should be describing the interfaces between key components of the database. It would be pointless to outline the implementation for each component in a database. Implementations change over time. If standards are developed for interfaces between components then databases will survive changes in implementation. From this, the study group developed a framework for modeling these interfaces. The complete framework is too complicated to be described in this summary. For a complete description see <ANSI>. A simplified schematic view is seen in figure 2-2 on page 17. (Note: The numbers representing the interfaces have been left consistent with those of the complete framework. Therefore, some of them do not appear in the figure.)

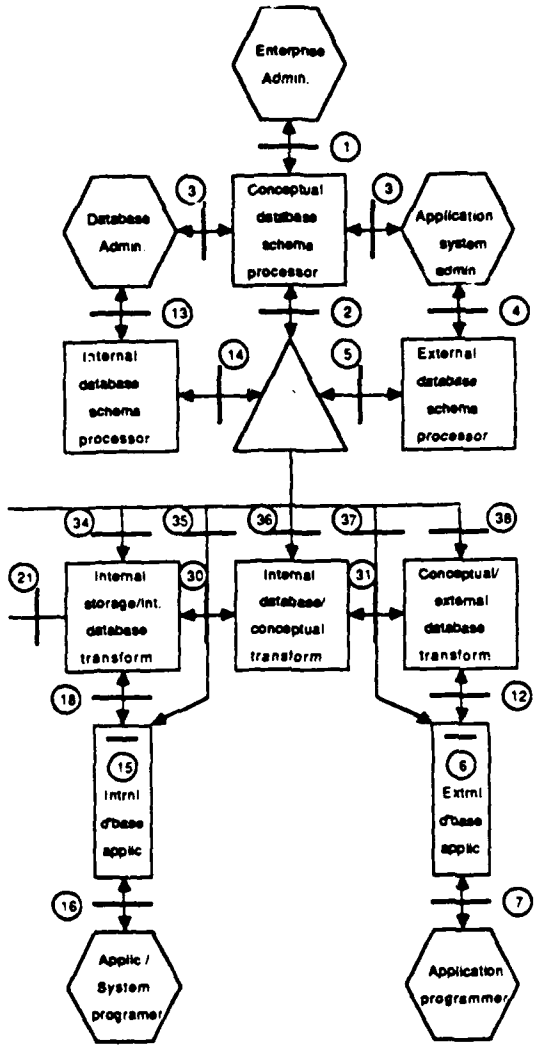


Figure 2-2: Partial schematic of ANSI framework.

In the figure, hexagonal boxes represent people in specific roles. The rectangular boxes are processing functions. The lines represent data flow, control information, programs, and data descriptions. Solid lines identify the interfaces between components. The dashed boxes indicate program preparation and execution systems. The triangle in the middle is representative of the database's data dictionary.

The diagrams specify function, not implementation. Each box may be representative of several other modules so long as the function of the box is the function described. In this modified framework the principle elements of the original framework are left intact. Also, interfaces are numbered in the same way as in the complete framework.

2.2.2 Framework concepts

The function of a database is to perform representation and manipulation of symbols about a limited part of the real world. This limited part is called the *enterprise*. There are three significant realms of interest in this area. The first is *external*. This is a simplified view of the real world as seen by one or more applications. The second realm is *conceptual*. This is the limited model of the real world as seen for all applications of the enterprise. The last realm is *internal*. This is a model of the data maintained for the representation of this limited model of the real world.

Within an enterprise there exist *applications*. An application is a part of the enterprise whose goal it is to accomplish a specific task in pursuit of the enterprise goal.

The external realm contains *external views* of the database, each of which is a collection of objects representing data of interest to a specific application. Each

external view is associated with an *external schema* describing the objects in that external view of the database.

There is also a *conceptual view* of the database, which is a collection of objects representing the data of interest to the enterprise. The objects are described in a description language. The description of objects according to this language is called the *conceptual schema.*

The internal realm also contains an *internal view* of the database. This is a collection of objects which are related to the objects in the conceptual view of the database. It is described by an *internal schema.* The internal realm is oriented towards the most efficient part of the computing facility. The internal realm is the point at which the database makes contact with the computing facility on which the database resides. Therefore, it must be easily modifiable. This means that if the particular computer on which the database is based is altered or replaced, the internal schema must be able to change to accommodate, thereby preserving independence of implementation.

The *data dictionary* contains all information about the database. This includes schema descriptions as well as descriptions of the mappings between schemas.

The *enterprise administrator* is responsible for creating the conceptual schema. He or she serves as the focal point for identifying information that is vital to the enterprise. He or she must also determine how this information is to be managed, as well as who should see it. In addition, he or she must describe the relationships between information objects. The conceptual schema is sent to the *conceptual schema processor* to be coded into a computerized form.

The *database administrator* must specify an internal description of the

information presented by the conceptual schema. The internal schema is then forwarded to the internal schema processor, which stores it in the data dictionary.

The *application administrator* develops an external schema based on the needs of the application programs utilized in the database. The application administrator is also responsible for control of all of the application programs.

2.2.3 The interfaces

The following describe the various interfaces required in the ANSI'X3 three-tier framework: (Please refer to figure 2-2 on page 17.)

Interface 1: This is the interface by which the enterprise administrator tells the database management system his declarations of the conceptual schema.

Interface 2: This interface presents the coded conceptual schema to the data dictionary for storage and retrieval.

Interface 3: The database administrator and application administrator use this interface to determine information about the conceptual schema.

Interface 4: This is the interface by which the application administrator lets the database management system know his intentions for the external schema.

Interface 5: This interface presents the coded external schema to the data dictionary for storage and retrieval.

Interface 6: This is the interface by which an external schema is made available for use in writing or processing an application program in the host language.

Interface 7: In this interface, an application programmer specifies the selection and manipulation requirements within the application program on external data objects defined in the external schema.

Interface 12: By this interface, an external data manipulation language is expressed in a form independent of any host language.

Interface 13: This is the interface by which the database administrator lets the database management system know his intentions for the internal schema.

Interface 14: This interface presents the coded internal schema to the data dictionary for storage and retrieval.

Interface 15: This interface presents the internal schema to those authorized to see it.

Interface 16: A programmer specifies access and manipulative statements on internal data objects defined in the internal schema using this interface.

Interface 18: By this interface an executing system program accesses and manipulates internal data objects that are defined in the internal schema.

Interfaces 34-38: These interfaces transmit the schemas and mappings to the various transform modules and program preparation and execution subsystems.

2.2.4 PDES: A standard that uses the framework as a model

PDES is the Product Definition Exchange Standard. The PDES project was started in 1984 and has two objectives. They are to "develop an exchange standard for product data in support of industrial automation [and] to represent the US position in the ISO and take the leadership in the development of a single worldwide standard for the exchange of product data." [<Kallel>]

The intention of PDES is to develop an exchange standard for product data as a whole. This means that the data should include all functional and physical characteristics of the manufactured product.

> It includes the geometry, topology, tolerances, relationships, attributes, and features necessary to completely define a component part or an assembly of parts for the purpose of design, analysis, manufacture, test, inspection, and product support.

[<Kallel>]

The information transmitted using PDES is presented to the receiving host in a form that can be directly used by an application program. The transmitted data does not change regardless of the architecture of the machine that it is being sent to. This is a radical departure from the Initial Graphics Exchange Standard (IGES). The purpose of IGES was to provide the exchange of data between individual systems.

The committee that developed PDES (ISO/TC184/SC4) based its approach on the ANSI/X3/SPARC three layer architecture.

> It aims to model the whole life cycle [of the product] using a formal data modeling methodology and integrate the different models of different applications into a single conceptual schema independent of a particular application view of the data and the technology used to implement it resulting in a common knowledge among different applications that is totally consistent between all different views.

[<Kallel>]

PDES is still in the development stages. It is a good example of typical standards of the future and is worth taking a closer look at when it is fully developed.

2.3 Summary

A database management system can be modeled by the many interfaces and modules shown here. However, if we are talking about connecting heterogeneous databases then we must also consider networking problems of the host computers running under the databases. This is where we can incorporate the two models presented here. This will be discussed in Chapter 3.

Chapter 3

One Model for the Integration of Heterogeneous Databases

The problem at hand is to achieve a single model for the integration of heterogeneous database systems. In Chapter 2, two models were presented that have been developed for similar purposes. They were developed to serve as an industry standard for networking (the ISO model) and as an industry standard for database management systems, respectively. It is of equal importance that we also develop a standard for integrating heterogeneous database systems.

3.1 The importance of one model as a standard

Standards setting in the computer industry is a big issue. As processors get faster, and microchips get smaller, new design decisions are made every day. Unless standards are created these design decisions can be radically different from one vendor's equipment to the other. For computer-based information systems to be complete we must "standardize at the semantic level all of the discourses which are essential to the design, installation, and operation of [them]." [<Bachman> p. 47]

These standards should capture elements of present-day machines as well as anticipated modifications to the current technology. "In the area of computers and communications there are two principle functions of standards: compatibility and variety reduction." [<Sirbu2> p.35] Compatibility functions are required for computers of dissimilar architectures to work together; whether for the purpose of resource sharing, information sharing, or electronic mail. Variety reduction concerns reducing the number of different versions of a product. The more

restricted a market is, the more similar its products will be. Variety reduction allows for a uniform conceptual point of view. Technology advances faster when all technologists *speak the same language.*

Like the two models presented in the Chapter 2, this model should not be concerned with the particular implementation of the network or the databases involved. Rather, it should be more general, so that it can connect to other networks based on the same model but implemented differently. The particular implementations chosen should be based on the underlying needs of the network, not on any generalized model. Another reason for this is that future implementations may be better suited for the network. Therefore, the network should be capable of adapting to new implementations. If the model is based on a particular implementation then it will not function properly when that implementation is changed. The implementation must be customized to the requirements of the system. This includes the requirements of both the programmer and the user. This is recognized by the U. S. Air Force: "The CALS [Computer Aided Logistics Support] architecture is ... unique do to the nature of the program." [<U. S. Air Force> p. 31]

The importance of having no single standard for implementation is also recognized by standards committees. For example, there exist several standards for representing graphics on computer systems. There is a modular family of graphics standards [<Cuthbert> p. 4]. This means that the implementor of a graphics system must recognize the needs of that specific system and choose the graphics standard appropriate for that application. One system may need a large set of textual fonts to be used with the graphics, while another might require less fonts but a wider range of text sizes. Different implementations should be chosen to meet the different needs.

In light of this, the importance of developing a single model for integrating heterogeneous database systems should be realized. It will serve several purposes. It will make it easier to connect existing systems. It will make it even easier to develop compatible databases in the future. Standardization efforts of the past have dealt with connecting heterogeneous computers in a network, and developing standard models for databases. Recent work in connecting heterogeneous databases necessitates development of a standard model. The next section will suggest the author's view of how this model should be developed.

3.2 Deficiencies of the ISO OSI reference model

If the concern is to model the interconnection of heterogeneous databases, then the models presented in Chapter 2 are of vital importance. Looking at each of the models from this perspective on an individual basis, several pitfalls arise. This does not mean that the models are faulty. It simply means that we cannot use either one of the models alone to model integrating heterogeneous databases. Reasoning for this is given below.

The goal of the ISO Open Systems Reference Model was "to make all programs, data, and other resources available to anyone on the network without regard to the physical location of the resource and the user." [<Tanenbaum2> p. 3] Its intention was to facilitate the interconnection of computers regardless of the applications being run on them. "Techniques for interconnecting networks depend on specific objectives." [<Pouzin> p. 241] This can also be applied to standards. The ISO OSI did meet its objectives. It did make it easier to develop networks of computers. There are some points that were overlooked by the model, however.

One deficiency of the ISO's Open System Interconnection model is that it

standardizes two different kinds of network services, rather than one. The two network services are called connection-oriented and connectionless. This is primarily due to the debate over datagram service versus virtual circuit service. The existence of these two standard services invites the unpleasant possibility of two incompatible communications environments in OSI. The information systems community has taken sides on this topic. This split comes from two different views of how networks should be put together.

The connectionless network service is based on experience with experimental networks, such as the ARPAnet. In this approach, each packet of data travels through the network independently. It must carry with it the information necessary to enable gateways to forward it correctly. Since the packets travel independently, the network layer does not retain any relationship between them. Packets may be lost, because of routing changes or congestion. In order to maintain data integrity, a protocol must be run over the network layer that will resequence or retransmit data.

The connection-oriented network service is more similar to a telephone network. It corresponds to the service provided by X.25 networks. Using connection-oriented network service, a connection must be established before data can be sent to a particular destination. Once the users and the network all agree on a connection, the data can be transmitted. The connection is broken after all necessary data has been sent. This process preserves data integrity.

The existence of the two of these services presents a standardization problem. The OSI Reference Model is designed such that particular implementation of protocols is not considered. However, because there are fundamental differences between connectionless and connection-oriented services, it would not be possible to transparently convert between them within the network layer. If both services exist

and are each accepted by different groups, then products will be provided which implement both. This would increase development and support costs, thereby defeating the purpose of a standard.

There are several reasons why the undesirable existence of two different network services is present in the reference model. When X.25 was developed in 1975, there was no reference model. However, the X.25 standard was the primary inspiration for the OSI model. So the principles of X.25 were *built-in* to the reference model. Therefore, the reference model had a bias towards connection-oriented service. Many corporations already have substantial investments in X.25 networks. To abandon them would be a costly endeavor.

Why would a company want to give up its investments in X.25? This is answered by the fact that connectionless service has more advantages. One advantage is that connectionless services are easier to build because of the stability of dynamic routing. Also, a transport layer protocol is always needed, and connectionless service requires no additional traffic and little additional complexity for the resequencing and retransmission functions. Another advantage is that connectionless service does not require the considerable memory, processing power, and network bandwidth necessitated by connection-oriented service. Finally, connectionless operation is more similar to the dynamic traffic patterns generally associated with computer networks.

Connection-oriented service does have the advantage that flow control can be more closely monitored then with connectionless service. However, the advantages of connectionless service outweigh this.

The large number of options within protocols is a severe problem with the OSI reference model. The result is that large numbers of potentially incompatible

combinations of protocols are created by different vendors prescribing to different beliefs. Even if products are built to handle all of the possible combinations, the complexity of the software involved would inhibit the reliability of the product. It is for this reason that I would endorse the use of connectionless network service in any network using the ISO open systems reference model as a basis.

Another problem with the reference model is that it does not give any standards for networking database management systems. It merely models connecting the machines on which these systems reside. It does not make any suggestions on how to connect the databases that exist on these machines. They all fall under the applications layer. This is to broad of a generalization to design a system after. Therefore, it seems feasible that a combination of the reference model and the ANSI framework for databases would be necessary for interconnecting heterogeneous databases. The next section will give reasoning as to why the ANSI framework cannot be used alone to model such systems.

3.3 Deficiencies of the ANSI framework for databases

When the ANSI framework was developed, interconnection of heterogeneous databases was not a topic of very much interest. The primary focus of the ANSI/SPARC study group was in local databases resident on one machine. Therefore, the model was developed with this in mind. It does do a good job of representing local database management systems. However, it does not touch upon the topic of connecting databases on different machines. Therefore, in achieving this goal it seems as if the framework should be combined with the ISO reference model.

3.4 A suggestion for a single model

The author suggests that a single model for the integration of heterogeneous database management systems should consist of a modified version of the ISO open systems interconnection reference model. The reference model should be expanded to allow for human-system interfaces. It should use the ANSI/SPARC database model as inspiration. The ANSI/SPARC database model (as described in Chapter 2) identifies three different human roles involved in database development. Similar human roles should also be identified in the expanded ISO reference model. Figure 3-1 represents the author's extended reference model for the integration of heterogeneous databases.

3.4.1 Identifying Human Roles

The human roles of a system can be described by the responsibilities that they hold. A set of responsibilities must be developed in order to identify the roles necessary for the development of an integrated database management system. These responsibilities should represent the tasks which must be performed for the system to function properly, both on the network level and on each individual host. These responsibilities must also take into account the objectives of database management systems. Another responsibility of these human roles should be to insure that hosts within an integrated system conform to the OSI reference model.

The human roles described next are represented in figure 3-1.

3.4.1.1 The System Administrator

It is quite clear that some kind of human management must preside over the entire integrated system. The role of this *system administrator* would include ensuring the reliable operation of the network, independent of the status of each

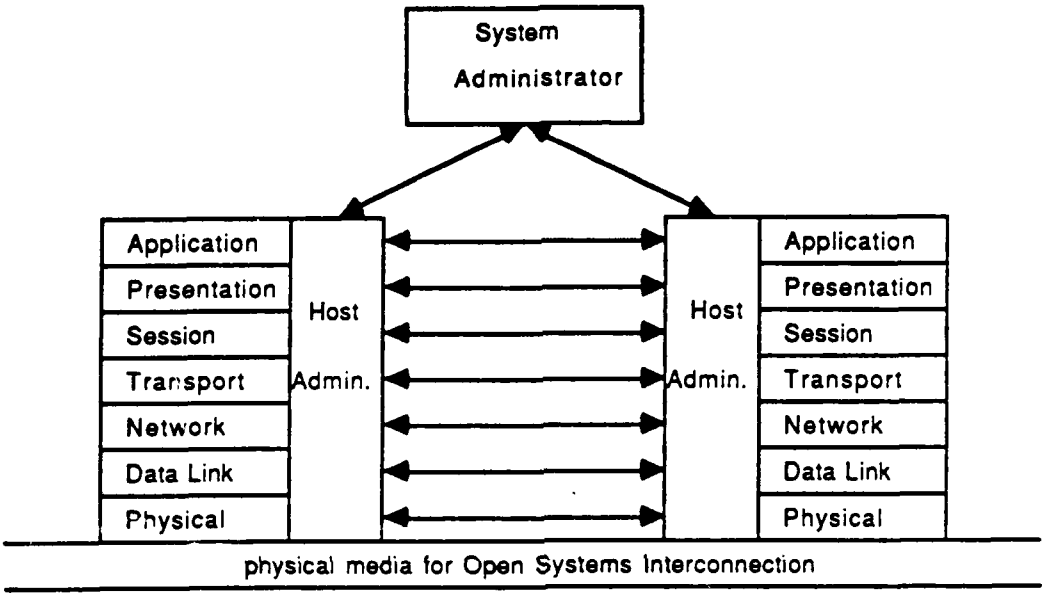


Figure 3-1: Extended reference model for integrating heterogeneous databases

individual host, at any given time. The system administrator should also be responsible for maintaining the objectives of database management systems between individual hosts.

The most important of these objectives is data independence. Data independence makes changes to representation, formatting, organization, modeling, or location invisible to the user. If data is to be exchanged between different hosts then the database structures of both the sender and the receiver must be

considered. While data independence does not include the capability of a database management system to automatically cope with such changes, it should be the responsibility of the system administrator to convey changes to hosts that are affected by the changes. Data independence does not mean that changes are to be avoided, but rather that a system be flexible to change. Change is inevitable. In the development of a system, changes to the data should be anticipated. The system administrator should maintain the system so that any changes can be accommodated for, without major effort.

A database management system that provides data independence ensures that applications can continue to run, perhaps at reduced performance, if the stored data is reorganized in such a manner that other applications will run with better performance. Such a database management system does not prevent one from rewriting and retuning the old application to take advantage of the new changes and enhance its performance.

The system administrator must also ensure that the system is capable of adapting to other types of changes. One such example is the addition of new hosts to the network. The system administrator must guarantee that new hosts can be added and must also manage the addition of such hosts. New hosts must be analyzed to see if they will be beneficial to the system, or if they will strain the system.

In order to guarantee the reliable operation of the system and that the objectives of database management systems are met, the system administrator may require some cooperation from each host in the network. There must be a human role at each host who is responsible for communicating with the system administrator.

3.4.1.2 The Host Administrator

The responsibilities of the *host administrator* are two-fold. The host administrator is responsible for communicating with the system administrator and for maintaining the reliable operation of the machine for which he is administrator.

The communication between the system administrator and the host administrator is represented in figure 3-1 by the arrow between the two roles. The host administrator must communicate with the system administrator in two respects. He must report any changes being made to his machine that would affect other hosts in the network. He must also identify any changes reported to him by the system administrator that will affect his machine. After identifying such changes he must ensure that the necessary modifications are made to accommodate for the changes.

In addition to this, the host administrator must be able to identify any changes to his database that could affect other users within the system. This should be done in cooperation with the system administrator.

The role of the host administrator is similar to that of the enterprise administrator in the ANSI/SPARC database model. He must identify information use within his database. He must also identify what level of security is to be used within the database, as well as the availability of the database to other users. The host administrator must develop his database in such a way that it is easy to adapt to changes to the system.

Another responsibility of the host administrator is to maintain consistency of his machine with the ISO OSI reference model. This is represented in figure 3-1 by the host administrator box overlapping all seven layers on each host. He must ensure that his machine is capable of interfacing with another at all levels of the

reference model. This means that he must be able to report the particular implementation of any of the levels to the system administrator of the network in such a fashion that a protocol can be established to make his machine compatible with others in the network.

3.5 Conclusion

A model has been presented here in an attempt to model the integration of heterogeneous databases. The author realizes that any system can be made to fit the mold of any model. However, when developing a model one hopes to obtain a balance between making broad generalities of all systems and being specific enough to make the interface between two hosts easier to construct. It is the author's view the model presented here achieves this balance.

References

[ANSI 78] ANSI/X3/SPARC/Study Group - Database Management Systems. *Framework Report on Database Management Systems.* Technical Report, American National Standards Institute, 1978.

[Bachman 82] Bachman, Charles W. and Ross, Ronald G. Toward a More Complete Reference Model of Computer-Based Information Systems. *Computers & Standards* 1, 1982.

[Brookes 82] Brookes, C. H. P., Grouse, P.J., Jeffery, D. R., and Lawerence, M. J. *Information Systems Design.* Prentice-Hall of Australia, 1982.

[Bussolati 81] Bussolati, U. and Martella, G. Access Control and Management in Multilevel Database Models. In Goos, G. and Hartmanis J. (editors), *Trends in Information Processing Systems.* 3rd Conference of the European Cooperation in Informatics, Springer-Verlag, Munich, October, 1981.

[Cuthbert 86] Cuthbert, Geraldine R. Ada WIS Foundation Technologies GKS ADA Language Binding. 1986.

[Gligor 84] Gligor, Virgil D. and Luckenbaugh, Gary L. Interconnecting Heterogeneous Database Management Systems. *Computer* , January, 1984.

[Kallel 87] Kallel, Maher. Standards for Data Exchange in an Integrated Environment: A Methodological Approach. 1987.

[Metz 86] Metz, Richard. Boeing's PC Practices. *Datamation* , January, 1986.

[Pouzin 77] Pouzin, L. Network Interconnection. *Future Networks - Infotech* , 1977.

[Roberts 70] Roberts, L. G. and Wessler, B. D. Computer Network Development to Achieve Resource Sharing. *SJCC* , 1970.

[Sirbu1 86] Sirbu, Marvin and Hughes, Kent.
Standardization of Local Area Networks.
Technical Report, Dept. of Engineering and Public Policy, Carnegie Melon University, April, 1986.

[Sirbu2 85] Sirbu, Marvin and Stewart, Steven.
Market Structure and the Emergence of Standards.
Technical Report, Dept. of Engineering and Public Policy, Carnegie Melon University, April, 1985.

[Sirbu3 84] Sirbu, Marvin and Zwimpfer, Laurence E.
Standards Setting for Computer Communication: The Case of the X.25.
IEEE Communications Magazine 23(3), 1984.

[Tanenbaum1 81] Tanenbaum, A. S.
Network Protocols.
ACM Computing Surveys 13(4), 1981.

[Tanenbaum2 81] Tanenbaum, A. S.
Computer Networks.
Prentice-Hall Inc., 1981.

[U. S. Air Force 86]
U.S. Air Force Plan for Implementation of Computer Aided Logistics Support
Headquarters, Air Force Systems Command, Andrews AFB, DC 20334-5001, 1986.

[Whitaker 87] Whitaker, William A.
A Commentary on the WIS Ada Foundation Technology Studies.
1987.

A TECHNICAL COMPARISON OF DISTRIBUTED HETEROGENEOUS DATABASE MANAGEMENT SYSTEMS

SUBHASH BHALLA, B.E. PRASAD,
AMAR GUPTA, AND S.E. MADNICK

The intent of distributed, heterogeneous database management systems is to provide a logically-integrated user-interface to physically non-integrated databases of several different types. This process of integration encompasses concerted retrieval of information as well as coordinated transaction management. Because of the added complexity involved in translating between multiple systems and multiple data models, distributed heterogeneous database systems are more complex than equivalent homogeneous ones.

In this technical report, eight different systems have been considered in detail. These systems and the respective developing organizations are as under:

(i) ADDS, Amoco Oil Company;
(ii) IISS, Air Force;
(iii) IMDAS, National Bureau of Standards;
(iv) MERMAID, Unisys;
(v) MRDSM, INRIA (France);
(vi) NDMS, CRAI (Italy);
(vii) MULTIBASE, CCA; and
(viii) PRECI, University of Aberdeen (Scotland).

Some of these systems are oriented for a particular computational environment, such as manufacturing, while other systems are intended to be general purpose. While all these eight systems are able to do global retrieves, their ability to perform global updates is varied. The salient features of all the systems have been summarized in a table. The report concludes with a list of areas requiring further research efforts.

TECHNICAL REPORT #5

A TECHNICAL COMPARISON OF DISTRIBUTED HETROGENEOUS DATABASE MANAGEMENT SYSTEMS

I. INTRODUCTION

The purpose of a distributed, heterogeneous database management system (DBMS) is to access, aggregate and update the information maintained in existing, distributed, heterogeneous DBMSs through a single uniform interface without changing preexisting (local) DBMSs and without disturbing local operations [GLI86]. To provide such services, within the constraints imposed by the existing set of heterogeneous local DBMSs, the critical aspects are as follows :

- Development of a Standard User Language and Data Model;
- Facilities for Query Processing;
- Incorporation of Distributed Transaction Management Routines; and
- Support of Distributed Operating System Functions and Network Services.

With the idea of gaining more insight into the above issues, we have put together various approaches being persued by researchers around the world. We have chosen a set of eight representative prototypes. The salient features of these eight systems are summarized in the following paragraphs.

Distributed Heterogeneous Systems.

II. OVERVIEW OF SAMPLE PROTOTYPE SYSTEMS

The prototypes of Distributed Heterogeneous Database Management Systems being assembled by various research teams differ significantly from each other in terms of their objectives and specific aims, and their design approach. Some of the prototypes are intended for a specific application, such as manufacturing, while others are general purpose. They also make dissimilar assumptions, such as some assume the existence of relational databases for future systems, whereas others deal with multiple types of databases.

The different systems, and their respective sets of objectives and assumptions, are summarized below.

MULTIBASE

MULTIBASE is a software system developed by Computer Corporation of America, Cambridge, Mass., for providing a uniform, integrated interface for retrieving data from several existing, heterogenous distributed databases [LAN 82], [SMI81]. The main objective of MULTIBASE is to answer queries. It allows a user to reference data in heterogeneous databases, through a common query language, using a single

database description [DAY83], [GOL84].

The integrated access available through MULTIBASE does not provide either the capability to update the data in the local databases, or the ability to synchronize read operations across several sites. To implement global concurrency control mechanisms for read or update operations, the global process must request and control specific services offered by the local systems (e.g., locking local items). MULTIBASE has no provisions for taking care of this type of activity.

The key objectives of MULTIBASE are: generality, compatibility and extensibility. It has been designed to be a general tool, without specific orientation towards any particular application area. It allows existing applications to operate without change. Also, it supports an easy extension path for adding new local systems to expand the existing MULTIBASE system configuration. The language provided to global users by MULTIBASE is based on the functional data model, and is called DAPLEX [SHI81]. DAPLEX provides a conceptually natural, database interface language. It uses constructs to model real world situations which closely match the conceptual constructs a human being might employ when thinking about those situations.

The process global retrieval involves two main

Distributed Heterogeneous Systems.

components. These are : Global Data Manager (GDM), and Local Database Interface (LDI). The functions of these two units are summarized in a table later in this report.

MULTIBASE employs a three level schema of definitions. In all, MULTIBASE provides an integrated scheme for

- Uniform query access to dissimilar DBMS's;
- Local schema integration;
- Data incompatibility handling;
- Local query optimization, and
- Global query optimization.

INTEGRATED MANUFACTURING DATA ADMINISTRATION SYSTEM (IMDAS)

The IMDAS architecture is an experimental facility, being implemented at the Automated Manufacturing Research Facility of the National Bureau of Standards. This testbed is intended to demonstrate the feasibility of supporting the manufacturing and production environment for factories of the future [BAR86], [LIB86]. The focus is on various functions related to manufacturing such as design, planning, and control. The main objective is to achieve a high level of software integration in an environment, consisting of

engineering workstations, robots, and other machines, each operating on an autonomous basis. Supplementary objectives include :

(a) Support for modular expansion, that is, support for Network reconfiguration

(b) Effective resource utilization,

(c) Efficient processing of time critical transaction and replication of data to support such activities.

Also, the goal of a Flexible Manufacturing System implies use of adaptive control techniques whereby the control system for such an environment is able to react to failures and unexpected events.

The data model used by the prototype for IMDAS is SAM*, which contains a variety of data semantics [KRI85], [SU 83]. It includes constructs for modeling the relationships among the data found in engineering, commercial, scientific and statistical databases. IMDAS supports three levels of new definitions, to describe the mappings between the single logical database and the multiple physically distributed databases. This characteristic of IMDAS is common to most heterogeneous distributed database management systems, though the control typology visualized for IMDAS is different from that of others.

Distributed Heterogeneous Systems.

In an Integrated Manufacturing System, a hierarchical pattern of events is visualized. At the top (the total facility) level, orders, process plans and part designs are entered. At the equipment or machine level, sensory information enters the system. A user or a control process can express a transaction in the Global Data Manipulation Language (GDML). The user process initiates a GDML query to the global external view of the integrated database. To process this query, IMDAS modifies the query tree so that the query operation operates on the data defined in the global conceptual view.

Between the centralized and distributed database management architecture, IMDAS has chosen a hybrid approach. IMDAS consists of three service layers, each of which is responsible for a definite set of distributed data management functions. These functions are distributed over the component systems according to their computational capabilities. The different layers of IMDAS software work together in establishing, manipulating, and controlling the distributed databases. For more details, please see table comparing all the prototypes.

Distributed Heterogeneous Systems.

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)

Integrated Information Support System (IISS) is a system sponsored by the Wright-Patterson Air Force Base. This system is being developed to support manufacturing and logistics environments for the U.S. Air Force [IIS83]. Its key design objectives are :

(a) Using common data available within various functionally independent subsystems through definition, control and execution of actions affecting information;
(b) Supporting information resource management of various application systems in a closed-loop environment within manufacturing; and
(c) Accessibility from geographically dispersed locations and support of future enhancements

The software integration is achieved by adopting a three schema approach. These are : (i) External Schemas (user views), (ii) Conceptual schema, and (iii) Internal schema. The conceptual schema employed is IDEF-Extended which is an Entity-Relationship based model. All the three schemata and the transformations among them are managed via

a three schema data dictionary, called the CDM (Common Data model) subsystem. The CDM dictionary, is maintained as a database that describes the conceptual schema and the network environment. This resource (CDM) is maintaied in a centralized fashion.

To process queries, query statements in DML within the global schema (conceptual schema) are embedded in COBOL and precompiled. On precompilation, source code files are sent to their respective hosts for compilation. IISS permits Integrated Application processes to occur along with Non-Integrated Application processes. In case of an Integrated Application process, a new application developed on IISS may access data which is distributed on several databases. A Non-Integrated application process may access a local DBMS for retrieval and update activity. It is likely that global update activity will be supported for Integrated Application processes.

IISS uses LAN and wide area communication to provide access to IBM 3081 (Network-IDMS), Honeywell level 6 (IDS/II), VAX (IDMS), and VAX (Relational - ORACLE). Further, a Kernel known as Network Trasaction Manager (NTM) has been implemented to provide sophisticated services on the network, such as interprocess communication through message passing.

Distributed Heterogeneous Systems.

PROTOTYPE OF A RELATIONAL CANONICAL INTERFACE (PRECI*)

A research prototype of a generalized distributed database system called PRECI* is being developed at the University of Aberdeen, in collaboration with a number of research centers, mainly in Britain [DEE85]. The system is fully decentralized, with both retrieval and update facilities, permitting hetrogeneous and existing databases to be specified as nodes. PRECI* is a research prototype within the PRECI project.

PRECI* is a generalized DBMS based on a canonical data model supporting relational, network and other data models as user views [DEE81], [DEE84]. It uses extended ANSI/SPARC architecture, its conceptual schema (called canonical schema) being written in a relational form. The principal data manipulation language includes an extended relational algebra called PRECI algebraic language (PAL), which offers a number of specific commands for data integration.

A nodal database in PRECI* is fully autonomous, with its independent nodal DBMS (NDMS) and nodal external schema (NES). The latter must provide a relational or PAL interface to the Distributed Database which uses PAL as the standard

Distributed Heterogeneous Systems.

language for communications. The PRECI* schema levels include:

- GES (global external schema) which supports user views.
- GDS (global database schema) which is formed by the collection of PSs.
- PS (participation schema) which describes nodal data with authorization controls
- NDS (inner) (nodal database schema)
- NDS (outer) (nodal database schema)
- NES (nodal external schema)

PRECI* permits participation of a node in one of the two ways :

(i) an inner node, which contributes to the Global Database Schema; and

(ii) an outer node.

If the number of nodes in a database is large, and the expected frequency of usage for some of these is low, then these nodes participate as outer nodes. This reduces the overhead of creation of the GDS and GES for catering to hundreds of nodes. Users are permitted to formulate queries through a suitable language for specific nodes.

Local data models supported in PRECI* are accessed via a relational (algebra) interface. The Local Database Schema

Distributed Heterogeneous Systems.

must be redefined to support relational algebra or PAL. PRECI* allows global updates on base relations only. If the data is replicated, update is performed only on the original copy and broadcasted to other copies.

A DISTRIBUTED DATABASE SYSTEM (ADDS)

ADDS is a software system being developed by Amoco Production Company. This system provides a uniform interface to existing heterogeneous databases which are resident on various nodes of a computer network [BRE84]. The conceptual architecture of ADDS is capable of integrating relational, network and hierarchical databases [BRE86]. For specific oil exploration and production projects, data is extracted from the IMS databases, sent to the project location, merged with the local data and stored locally in relational and pseudo-relational databases, as a part of regular data extraction and data merge operations. The user is provided with a relational view of the integrated database and can formulate queries using relational algebra operations over the predefined set of relations.

The structure of the ADDS databases includes:

Distributed Heterogeneous Systems.

- Physical Databases (PDBs) which are databases that actually exist on a computer network node.
- Logical Databases (LDBs) which are Database Management Systems for the associated PDBS.
- Composite Database (CDB) which contain a collective view for a set of LDBs, that constitute a single database, from the designer's point of view.

The CDB has a centralized ADDS directory, where ADDS schema definition and CDB information is recorded. The directory is maintained as a relational database allowing the user to interactively access the directory and become familiar with the various CDBs available for processing.

A query in ADDS is addressed to the CDB, which translates it into subqueries addressed to local LDBs. Each subquery is translated from the ADDS query language into the query language and/or transactions of a specific DBMS.

In addition to the data definition information, the directory also contains the information used by the query optimization process. The use of relational structures for the directory provides flexible tools for maintaining the

directory. The query language provides the user with not only a universal view (a relation of logical fields), but also with a relational view, which expresses the CDB as a set of PDBCs and their logical fields. The reason for having seperate relational and universal views of CDBs is to provide a range of query capabilities for users of varying sophistication.

MULTICS RELATIONAL DATA STORE MULTIBASE (MRDSM)

MRDSM generalizes the MRDS relational database management system of HONEYWELL, to support multiple databases. MRDSM is being developed by INRIA (France). It operates on a specialized domain of multiple MRDS relational databases running on HONEYWELL systems [LIT85], [LIT86], [WON84]. It is not a true heterogeneous system. Heterogenity is dealt at the semantic level by providing uniform access to all databases implemented with the same DBMS. The query language is MDSL, which is SQL like. This language is an extended version of DSL which is the data manipulation language for MRDS.

A Global Schema does not exist in MRDSM as users can create conceptual schema known as multischema with elements

Distributed Heterogeneous Systems.

from local database schemas. Multischema is also associated with one or more Dependency Schemas to handle inter database dependencies.

A query on multischema is decomposed into queries on local databases after removing inter-data dependencies that cannot be handled locally. Then a working Database(DB) Schema is created to collect data from different databases. The collection process has been optimized. Finally queries are generated on the working DB to combine data togather.

ARCHITECTURE FOR INTEGRATED DATA ACCESS (AIDA / MERMAID)

MERMAID is an integrated data access system being developed by System Developement Corporation [TEM86b]. It allows users of multiple databases (relational DBMSs), running on different machines to manipulate data using a common language, which is either ARIEL or SQL [MAC85], [TEM86a], [YU 85].

The major processes of the MERMAID system are as under :

(a) The User Interface Process: It contains an embedded ARIEL or SQL parser and a translator that produces DIL(Distributed Intermediate Language).

(b) The Distributor Process: It contains an optimizer

and the controller.

(c) One DBMS Driver Process for each database to be accessed: This driver also contains a translator from DIL to the DBMS query language.

All information about schemata, databases, users, host computers, and the network is contained in a DD/D(Data dictinary/Directory) which is centrally stored in a database and accessed through a special driver. The translator and optimizer access the DD/D in order to do translation and query planning.

To process a query prepared by a user using ARIEL or SQL, the translator parses and validates the query and passes it to the distributor. The controller part of the distributer reads the query in DIL and passes it to the optimizer part which plans the exicution. The DIL query may need to be decomposed into several subqueries and the controller sends them to one or more DBMS drivers for execution.

The DD/D contains information about the databases, the users, the DBMSs, the host computers, and the network. It supports the following four layers of schema definitions.

(a) Subschema layer: It represents the users view based on the global schema.

(b) Global schema: It contains the federated view of

Distributed Heterogeneous Systems.

all the data definitionsin the distributed global schema.

(c) Distributed local schema: It represents the relational view of the local schema.

(d) Local schema: This schema corresponds to the external view of the local database.

The MERMAID integrated access system has been implemented using VAX (IDM, Britton-Lee), SUN 170 (INGRES), SUN 120(INGRES), and SUN 120(MISTRESS). Presently the system permits updates to a single database on an individual database basis.

MERMAID is an operatinal prototype which demonstrates the feasibility of operating as a front-end to distributed heterogeneous databases. A schema design tool is being developed which supports the user in developing the global view of the database from an existing schema.

NETWORK DATA MANAGEMENT SYSTEM (NDMS)

NDMS is a system being developed by CRAI (Italy) for the National Transport Informatic System of Italy [STA84].

NDMS supports the relational data model. The relational data model is supported as a view over various hetrogeneous

data models, namely network, hierarchical and relational data models. Relations pertaining to the respective levels of view definition are materialized during query processing. The view definition is organized hierarchically as a series of data abstractions. The three distinct abstraction levels are the NDMS Internal Schema, the Application schema and the End-user Views.

The NDMS Internal Schema comprises of base relations defined as aggregations over the local database Schemata. The base relation definitions require data mappings to be specified for each local database.

The NDMS network consists of logically interconnected nodes. The NDMS node comprises of the System Encyclopedia and the NDMS control software. The System Encyclopedia contains all information pertaining to the respective node, that is, user definitions, database mapping definitions, transaction definition etc., and the complete NDMS Internal Schema Definition.

The Node Data administrators responsible for NDMS applications at each node define relational views, using the SEQUEL view definition mechanism, as a collection of data abstractions (aggregations and generalizations) over the NDMS internal schema. The NDMS version of SEQUEL has been modified to handle the generalized abstractions. Defined

relational views are available to the end-user for defining their specific data abstractions.

The two basic transaction types supported by the NDMS are the queued transactions and the on-line transactions. The queued transactions are processed as local or remote batch processes. No exchange of messages is permitted for such transactions. The on-line transactions are considered to be distributed transactions. The NDMS Transaction Processor provides facilities to invoke transaction programs, to support the user interface, to exchange messages between application programs, and to synchronize transaction commit operations. A System Journal to support recovery mechanism exists.

We have described the major characteristics of eight different efforts in the area of distributed heterogeneous DBMSs. These eight systems were chosen on the basis of their uniqueness and the level of technical information available about these systems.

Distributed Heterogeneous Systems.

III. COMPARISON OF SYSTEMS

In this section, we compare and contrast the different approaches adopted by the eight representative systems.

A. UNIFORM INTEGRATED ACCESS IN DISTRIBUTED HETEROGENEOUS DBMS

The task of providing Integrated Access for a Distributed, Heterogeneous DBMS, involves providing a Standard User Language and a Data Model. In addition, query processing and query optimization in a distributed environment also need to be incorporated.

The issue of providing a standard user language and a standard data model is related to global data administration. The basic components of integration are the local DBMSs existing at user installations, which could be as varied as large mainframe based hierarchical-IMS at one end and personal computer based DBASE-II relational database on the other. The eight prototypes are geared to support different sets of local data models. MULTIBASE, IISS, and NDMS support DBMSs with relational and network data models [GOL84], [IIS83], and [STA84]; IMDAS supports only

relational local data models [LIB86]; ADDS supports both relational and hierarchical local data models [BRE84]; and PRECI* supports any model via relational algebra interface [DEE85]. Finally the two remaining prototypes MERMAID and MRDSM also support relational data models and other data models through a relational interface.

In order to provide a uniform integrated access to a system of heterogeneous DBMSs, a hierarchy of three functional layers has been proposed [GLI84]. These three layers are :

(a) The Global Data Management Service

(b) The Distributed Transaction Management Service

(c) Network Services.

The global data management (GDM) provides services directly to the end user, as the top most layer. The functions associated with GDM include (i) providing the global data model supporting a global schema, which is the basis for both the distributed DBMS user's view of the data and the standard user language; (ii) query decomposition; (iii) query translation; (iv) execution plan generation; and (v) results integration.

The global data model mentioned above needs to capture

the complete meaning of information stored at various Distributed Databases, due to this constraint it is usually defined by an entity relationship or semantic data model [APP85]. It is also referred to as the Conceptual Schema. All the data in the environment are defined in global conceptual schema, which is mapped to many underlying file and DBMS structures (referred to as internal schemata), and many user views (referred to as external schemata). Most prototypes for the Distributed, Heterogeneous DBMS adhere to this three-schema approach to data integration. All the three layers mentioned above fall under the application layer of the ISO reference model [GLI84].

The current prototype of MULTIBASE supports a Functional Data Model with an associated Data Manipulation Language (DML), DAPLEX [SHI81]; IMDAS uses Semantic Association Model (SAM*) and a SQL like query language [SU 83]; IISS supports IDEF-Extended which is Entity-Relationship based and queries are embedded in COBOL and precompiled [IIS83]; PRECI* supports a Canonical Data Model with PAL (PRECI Algebraic Language) which is relational algebra based [DEE84]; other prototypes such as MERMAID, ADDS and NDMS support Relational Global Data Models with MRDSM using an extended relational global data model, the

associated query languages for these prototypes are SQL like or relational algebra based [TEM86b], [LIT85], [BRE86], [STA 84].

The above data models are supported by very powerful data definition languages (DDLs) that are used to define the global conceptual schema in terms of objects, events and also to specify integrity constraints on the relationships and dependencies [APP 85]. The conceptual to external (user view) transformation is achieved through a global DML, that is similar to relational algebra or extended relational algebra as described above for most of the prototypes. The conceptual schema to internal schema is the other transformation required. This involves translation of both structure and form for all the heterogeneous DBMSs included in the system. Usually transformations of this type are performed by software at each node and data are moved through the network in the conceptual schema form. The three levels of schemata for one of the prototypes, IMDAS, being developed for automated factory environments, is shown in Figure 1.

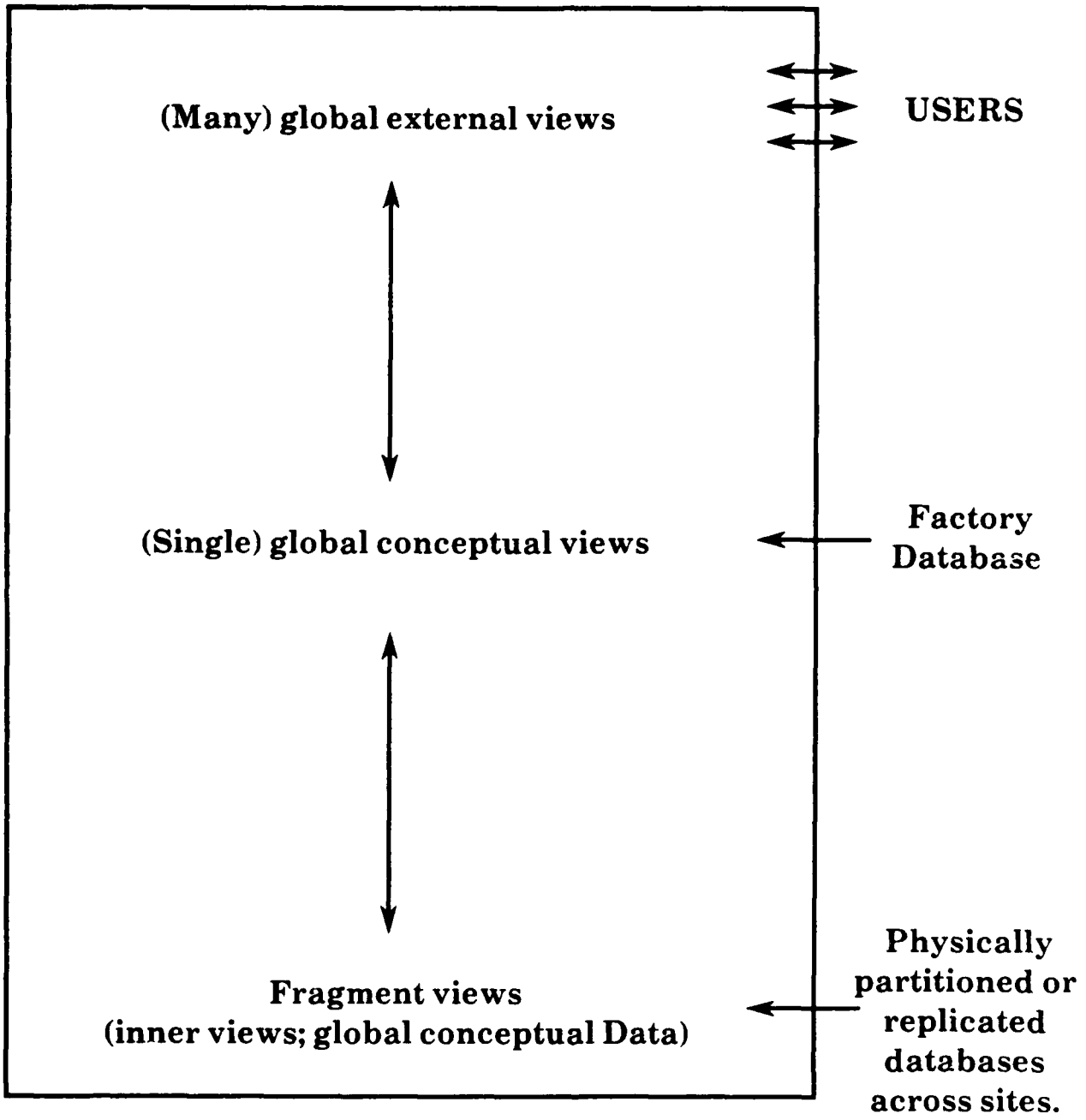


Figure 1: DBMS architecture for IMDAS, prototype distributed, heterogeneous DBMS for factory automation.

Distributed Heterogeneous Systems.

All the three schemata discussed above and the transformations between them are managed via a three-schema data dictionary. The existing databases use a two-schema data dictionary which is designed for individual DBMSs only. Most researchers are building their own three-schems data dictionary, e.g., IISS has a CDM (Common Data Model) subsystem [IIS83]. The CDM submodel consists of two software modules : (i) the CDM dictionary, which is a database that describes the conceptual schema and the network environment, and (ii) the CDM processor which is the software, that accesses the CDM dictionary and transforms user's data requests into transactions that can be processed by the local DBMSs. The DM processor is the distributed database manager of IISS. For further discussion on Local Database Schema conversion, Data incompatibilities and semantic mismatches and global schema construction, please see table comparing all prototypes. Most prototypes maintain the data dictionary as a centralized resource at one site and all distributed database managers access this central data dictionary.

QUERY PROCESSING AND QUERY OPTIMIZATION

The global query is fragmented into sub-queries by a

query decomposer. It is a function normally performed by a Global Data Manager (GDM), which uses the distributed (or centralized, for some of the prototypes) data dictionary as a guide. The query decomposition strategy of heterogeneous systems does not differ from that of homogeneous systems [GLI84]. However, in the case of heterogeneous systems , a language-to-language translation is required to mitigate the problem of data model differences. The query processing steps for MULTIBASE systems are shown in Figure 2. These include : language interface, global-to-local translation, query decompostion, sub-query translation, execution plan gencration, sub-query results interpretation and result integration [DAY83].

Query optimization considers consideration of processing queries and intersite processing. The Distributed, homogeneous DBMS such as R* and SDD-1 provide useful models for processing queries and query optimization [APE83], [BER81], [DAN82], [YU 83].

B. DISTRIBUTED TRANSACTION MANAGEMENT.

Distributed Transaction Management (DTM) as a function involves controlling the execution of distributed

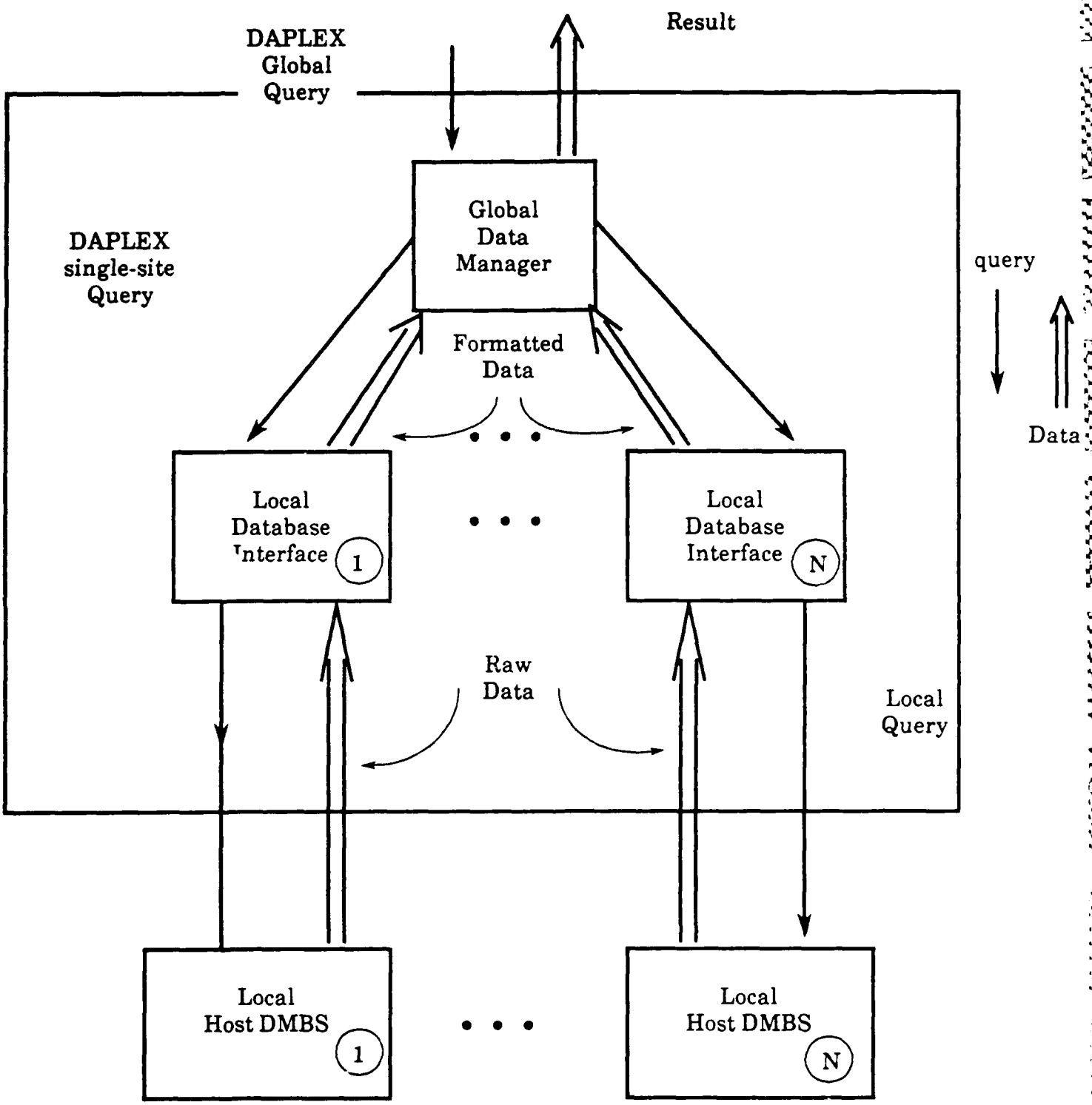
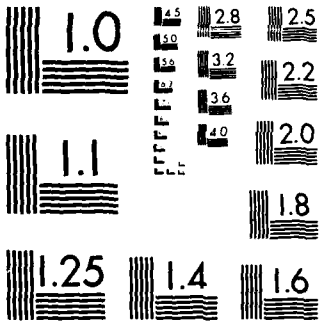


Figure 2: MULTIBASE query processing.

AD-A195 852 INTEGRATING DISTRIBUTED HOMOGENEOUS AND HETEROGENEOUS 3/3
DATABASES: PROTOTYPES VOLUME 3(U) MASSACHUSETTS INST OF
TECH CAMBRIDGE A GUPTA ET AL DEC 87 MIT-KBIISE-3
UNCLASSIFIED DTRS57-85-C-00083 F/G 12// NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

transactions in a Distributed, Heterogeneous environment. The software system responsible for providing DTM services must ensure that the consistency of the common shared data is preserved, in view of the possibility of multiple transactions accessing the same set of data as well as the possibility of site failures. To achieve these objectives, it becomes necessary to incorporate concurrency control and recovery procedures into the environment [GLI84]. This is in contrast to the case of Distributed Homogeneous systems, where the problems of integrating concurrency and recovery procedures does not arise.

Most prototypes presently provide a retrieve only interface to the Distributed, Heterogeneous DBMS. PRECI* allows global updates to local DBMSs. In case of replicated data, update is performed only on the original copy and broadcasted to other copies. Prototypes such as IMDAS, IISS, ADDS and NDMS are committed to solving the update problem.

The possibility of providing an additional layer of a software system to cater to concurrency control and recovery, without disturbing the existing heterogeneous DBMSs, has shown some promise [MAD87]. In this approach, transactions are classified as, those which issue updates

for other sites, and those which execute at the sites where updates are required to be made. The DTM service interacts with local DBMSs to implement the desired update activity. Not much is known about the approaches being taken by the research groups engaged on concurrency control and recovery for the prototypes being discussed.

C. NETOWRK SERVICES

The problem of interconnecting resources in a heterogeneous environment can be tackled by adopting one of two alternatives:

(1) Share services via a 'loosely-coupled' network, and

(2) Share resources via a 'transparent' network.

In the case of a loosely-coupled network the basic facilities supported are remote procedure call/message passing, naming and access control. In addition there are services like filing, mail transfer, and remote computation. Most current prototypes for Distributed, Heterogeneous DBMSs use network services that belong to this category.

In the case of transparent network services, distributed operating systems such as CRONUS developed by Bolt Beranek and Newman, Inc. provide a level above host

operating systems [SCH85] that appears to be a distributed operating system to the application programs. This level supports operating system functions including communication, access control, naming, and data storage and retrieval [TEM86a] [TEM86b]. This concept is also used in IISS which offers an Operating System component called NTM (Network Transaction Manager). The NTM is stated as an operating system above the existing operating systems and supports the application programs.

The Integrated Software architecture can be implemented on one of the two network-services alternatives described above. Most of the systems use services of existing type of the loosely-coupled networks or build a layer of services on top of the existing network services. For more details on research prototypes of Distributed Operating Systems, please refer [TAN85].

IV. KEY CHARACTERISTICS

The major features of all the eight prototype systems are summarized in the following table. This table consists of two parts. Part I contains details of four systems, and Part II provides information regarding the other four systems.

DISTRIBUTED HETEROGENEOUS DATABASE SYSTEMS PROTOTYPES

PART I

	MULTIBASE	IMDAS	IISS	PRECI*
ACRONYM DERIVED FROM	None	Integrated Manufacturing Database Administration System	Integrated Information Support System	Prototype of a Relational Canonical Interface
ENVIRONMENT	General	Manufacturing	Manufacturing and Logistics	General
ORGANIZATION	Computer Corporation of America (US)	National Bureau of Standards (US)	Air Force (US)	University of Keele (UK)
LOCAL DATA MODELS SUPPORTED	Hierarchical, Relational, Network	Relational	Relational, Network	Any model via relational algebra interface
LOCAL DBMS SUPPORTED (Current Version)	CODASYL database, Hierarchical database	Relational - INGRES, RIM, and (DB2 planned Oct. 85)	Network - IDMS on IBM 3081, IDS/II on Honeywell level 6, IDBMS on VAX, Relational - ORACLE on VAX	Under Development
GLOBAL DATA MODEL	Functional Data Model	Semantic Association Model (SAM*)	$IDEF_1$ (ER Based)	Canonical Data Model
GLOBAL DATA MANIPULATION LANGUAGE	DAPLEX. Local host may not support all capabilities provided by DAPLEX.	SQL-like. Supports interactive access and programs through attachment to local interprocess communication	No interactive query language. Query statements embeded in COBOL and precompiled.	PAL (PRECI Algebraic Language) which is relational algebra based

	MULTIBASE	IMDAS	IISS	PRECI*
LOCAL DATABASE SCHEMA CONVERSION	Local database schema called Local Host Schema (LHS) in any model needs to be redefined completely in Functional Data Model to enforce uniformity. New schema is called Local Schema (LS). Local host schema remains intact.	Local database schema redefined/extended into relational model with system utilities.Mapping work which would leave local host schema intact is progressing.	Redefined using Neutral Data Definition Language (NDDL). (No information about the structure of NDDL is available. It has been mentioned that it is capable of supporting relational and network schemas , entities and relations, and mapping.)	Local database schema must be redefined to support relational algebra or PAL.
DATA INCOMPATABLITIES AND SEMANTIC MISMATCHES	Resolved through an Integration Database Schema (called IS) for all the Local Schemas. Extent of coverage is not sufficient, but it can take care of differences like kilometers and miles, age in years and in qualitative form, (young, old), etc.	Performs conversions, maintains relationships between dependent relations and parent ralations. Resolved by Global Schema. Mapping Provided by Global Schema.	Performs data format and unit conversion operations.	Local database schemas converted to the relational form are placed directly in Global Schema. Integrartion data also placed separately in Global Schema . Unlike MULTIBASE, users can refer to incompatible information (e.g., kilometers and miles) through Global External Schema separately rather than as one unit. Mapping is provided by Global Schema.

	MULTIBASE	IMDAS	IISS	PRECI*
GLOBAL SCHEMA CONSTRUCTION	Uses the Functional Data Model to define the global schema and the associated query language DAPLEX as the global data manipulation language. The local DBMS and their Local Host Schema (LHS) are mapped into local schema (LS)which are expressed in terms of the Global Data Model. The local schema are merged into global schema with an auxiliary Integration Schema (IS) which contains information needed for reconciling inconsistencies between the local schemata.	On line SAM* Dictionary utilities for defining Global Schema, Views, Site capabilities, Fragmentation, and Partitioning.	Since the structure of NDDL is unknown, it is not clear how various local schemas, redefined in NDDL are merged.	Converted Local Schemas in relational form are placed in Global Schema with node information. Integration data and meta-data are maintained in relational form in the Global Schema. Merging of relations and related mappings are maintained in Global External Schemas (Views).

	MULTIBASE	IMDAS	IISS	PRECI*
QUERY PROCESSING (Retrieval)	A Query expressed in DAPLEX on Global Schema is decomposed into Queries on DAPLEX Local Schemas. These are further translated into Local DML queries. Adopts "no two variables in a query range over the same entity type" strategy to generate queries on Local Schemas. Adopts data reduction and data movement techniques in Global Query optimization strategy. Merging and final processing takes place at Global Schema result node.	A Query on Global External View is modified and converted to queries on Global Schema. If Global Schema Manager (called DDAS) cannot process the query, it is passed to the Master Schema Manager (called MDAS) for further processing; otherwise, it is translated into queries on local schemas.	Currently, query statements in Neutral Data Manipulation Language (NDML) on Global Schema are embeded in COBOL and precompiled. On precompilation, source code files are sent to their respective hosts for compilation.	This system follows the same strategy as MULTIBASE except in the final processing. The designers have aimed at a higher degree of parallelism. While in MULTIBASE, final processing is done at a result node, PRECI facilitates parallel processing at multiple nodes.
GLOBAL UPDATE AND TRANSACTION MANAGEMENT	Not Supported (Further work suspended due to lack of funds.)	Contains a component called Transaction Manager, to handle integrity, concurrency control and recovery. Allows Global update on base relations only. Recovery limited to cancel/restart. More work is in progress.	Committed to supporting global updates.	Allows Global update on base relations only. If data is replicated , update is done only on the original copy and broadcasted to other copies. (Not fully implemented as yet)

	MULTIBASE	IMDAS	IISS	PRECI*
NETWORK SERVICES	No Information	Token bus, Ethernet LAN. TCP/IP,HDLC have plan to use MAP Compatable network. Inter- process communication through shared memory. Transactions + Data encoded in ANS.1 (ISO 8824/8825 standards)	LAN and wide area communication. A kernel known as Network Transaction Manager (NTM) has been implemented to provide sophisticated services on the network. Interprocess communication through message passing.	Designers propose to develop a standard protocol for communications between distributed databases.
CURRENT STATE LIMITATIONS AND FUTURE DIRECTIONS	Internal prototype ready. U.S. Air Force has showed an interest in using it on an experimental basis in their applications. Currently permits READ-ONLYaccess and local updates.	Global Schema Manager (called DDAS) and local schema manager (called BDAS) exist on UNIX and VMS. Work is in progress on Master Schema manager. Research is on for coordination and synchronisation of distributed transaction management functions.	Feasibility demonstrated using a scenario selected by 22 contractors. Further work on interactive query facility and global update is pending.	Basic design is complete. Currently implementing a pilot system. Initial plan is to have two nodes at Aberdeen on Honeywell, one node at Belfast on VAX, and a fourth in Dublin on another VAX.

	MULTIBASE	IMDAS	IISS	PRECI*
REMARKS				Designed to handle replication of databases.
CONTACT PLACE (Person)	Computer Corporation of America Four Cambridge, Center Cambridge, MA 02142 (D. Smith)	Integrated Systems Group, Factory Automation Systems Division, National Bureau of Standards, Gaithersburg, MD (M. Mitchell, E. Barkmeyer , D. Libes or Howard Bloom)	Material Laboratory, Air Force, Wright Aeronautical Labs, Air Force Systems Command Wright-Patterson AFB, OH 45433	Dept. of Computer Science, University of Keele, Staffs. ST5 5BG, England (S. M. Deen)
REFERENCES	[SMI81, LAN82, GOLD84, DAY83, SHI81]	[LIB86, BAR86, KRI85, SU83]	[IIS83]	[DEE84, DEE85, DEE81, DEE87a, DEE87b]

DISTRIBUTED HETEROGENEOUS DATABASE SYSTEMS PROTOTYPES

PART II

ACRONYM DERIVED FROM	None	Multics Relational Data Store Multibase	Amoco Distributed Database System	Network Data Management System
ENVIRONMENT	General	General	General (mostly scientific and engineering databases relating to data connected with Oil Wells)	Transportation
ORGANIZATION	UNISYS, Formally System Development Corporation (US)	INRIA (France)	Amoco Production Company, Research (USA)	CRAI (Italy)
LOCAL DATA MODELS SUPPORTED	Relational or Relational *Interface*	Relational or Relational Interface	Relational, Hierarchical and Network	Realational and Network
LOCAL DBMS SUPPORTED (Current Version)	IDM (Britton-Lee) on VAX, INGRES on SUN 170 and SUN 120, MISTRESS on SUN 120, M204 under development.	Multiple MRDS on Honeywell	Hierarchical - IMS, INQUIRE Relational- SQL/DS, RIM, FOCUS; Some sequential file formats	Network- IDMS DB/DC and ADABAS on IBM; Relational - RODAN on IBM, INGRESS on VAX
GLOBAL DATA MODEL	Relational	Extended Relational	Relational	Relational
GLOBAL DATA MINIPULATION LANGUAGE	SQL or ARIEL (SDC query language)	MDSL SQL like, extended version of DSL (the data manipulation language of MRDS)	Extended Relational Algebra Language, plus a subset of the ANSI SQL language	Modified version of SQL

	MERMAID	MRDSM	ADDS	NDMS
LOCAL DATABASE SCHEMA CONVERSION	Needs conversion into relational form. (System does not provide help for conversion from local data model to relational form or from relational form to local data model), Can map user relational view to repeating groups in local database.	No conversion required. The system is designed to serve databases implemented with MRDS DBMS on Honeywell machines. The aim is to deal with semantic heterogenity in order to provide uniform access to these databases.	Needs conversion into relational form. The schemas of the local databases are described as Physical Databases (PDBs). PDBs are comprised of Physical Database Components (PDBCs). A PDBC for a relational database is a relation. A PDBC for a hierarchical database may represent a path from the root segment of the database to a leaf segment.	Needs conversion to relational form. Has two parts: a) Base relations; and b) Semantic information mapping. Base relations are constructed with objects from underlying database schema. Semantic and manipulation information about these objects is represented with Mapping Definition Language.
DATA INCOMPATABLITIES AND SEMANTIC MISMATCHES	Deals with two types of data translation schemes: functional type and enumerated type. Functional translation deals with problems like unit conversions (e.g., kilometers and miles) and format conversions (e.g., date) etc. Enumerated type translation deals with converting sets of values through a table lookup (e.g., codes and names).	Handles three types of interdatabase dependencies. They are a) manipulation dependencies; b) privacy dependencies, c) equivalence dependencies. Equivalence dependencies handle data incompatabilities and semantic mismatches.	Supports different names and characteristics (e.g., data types and units) assigned to semantically equivalent fields located in different databases through the system data definition language. Data conversion is performed as the physical data is loaded into the logical temporary relations.	Maintains information along with Global Schema in System Encyclopaedia. Further details are not available.

GLOBAL SCHEMA CONSTRUCTION	All the relations from different local schema are grouped through a schema translation mechanism. Schema translation deals with name changes, partitions, one-to-many and many- to-one mapping of relations (e.g., A global relation 'ship 'can be mapped to a relation with *name 'boat' in one* database and a join of two relations in another database). Local database schemas subject to schema and data translation mechanisms are maintained in Global Schema in relational form.	Global Schema does not exists. Users can create conceptual schema known as multi schema with elements from local database schemas along with one or more dependancy schemas to handle inter database dependencies. The elements of multischema created by *the user are :* (i) one or more dependency schema, or (ii) explicit enumeration of local databases to compose a multidatabase.	A single Global Schema does not exist. Instead, a Composite Data Base Schema (CDB) definition is used to describe the subset of all the physical databases that may be appropriate for an application. Many CDBs may be defined, depending on the number of applications. A CDB is comprised of *one or more* Logical Data Bases (LDBs) and an LDB is comprised of one or more Physical Databases (PDBs). An LDB is the set of PDBs that are managed by a single DBMS at a single site. All the CDBs are placed in a directory. A user can choose a CDB, define a view and query on this view. Also, a user can define his own CDB schema.	Local database schemas converted in relational form are further analyzed for naming conflicts and type conflicts. Equivalent base relations are integrated together.

	MERMAID	MRDSM	ADDS	NDMS
QUERY PROCESSING (Retrieval)	A query in SQL or ARIEL on Global Schema is translated into queries in an intermediate language called DIL (Distributed Intermediate Language), and later translated from DIL to the Local DBMS language. Currently translators are available for IDL, QUEL and SQL. Query Processing supports fragmented and replicated relations. Final post processing is done by one of the hosts.	A query on multischema is decomposed into queries on local DBs after removing interdata dependencies that cannot be handled locally. Then a working DB is created to collect data from different DBs. The collection process is optimised by performing projection and selection operations on source DBs. Finally queries are generated on the working DB to combine data.	Queries can be submitted against any number of Composite Data Bases (CDB). A query is optimized for minimal communication cost and a subquery schedule is produced. The steps of the subquery schedules are processed by server processes. A subquery against a base relation is translated into the language of the local DBMS.	An Application Schema is defined at each node over internal schema that represents a group of users at that node. Users at that node define their views on this Application Schema. Queries posed on Application Schema are translated to local database language through Internal Schema. Query processing strategy has three parts: the intermediate storage structure, query optimization and query execution.
GLOBAL UPDATE AND TRANSACTION MANAGEMENT	Update to single database is permitted.	Limited Multidatabase Update facility exists.	Conducting studies on the multidatabase update problem. ADDS retrival system is being modified to evaluate new concurrency control algorithm.	Updates are done through co-ordinated distributed processing applications. Transaction programs need to be implemented as application programs of the host DBMS. Co-ordinated by Transaction Processor.

NETWORK SERVICES	Ethernet, TCP/IP. Plan to use MAP and DODIIS prtocols for interprocess communication.	The system uses inter Multics communication facilities over X.25 TRANSPAC Net (French national net), using Dial-out interface	SNA, Ethernet (TCP/IP), Bisync. Uses a Logical network approach to provide a common interface to the physical networks.	X.25 protocol. Inter process communication through message passing.
CURRENT STATE LIMITATIONS AND FUTURE DIRECTIONS	Not a commercial Product. In use within DOD systems developed by UNISYS. Operates only on top of multiple relational DBMS. Tested on a Navy Database that contains ships, positions, weapons installed on the ships, etc. The database is distributed to four sites. Plan to extend into the areas of object management, security, and integration with deductive inference engine.	Investgating application of knowledge processing techniques in the area of query processing , to simplify query expressions.	Current version operates under VM/CMS and accesses data controlled by IMS, SQL/DS, RIM, FOCUS, and INQUIRE databases. The current version is being enhanced. The system includes a remote user interface and programming language interface for use on workstations. Future directions include supporting update transactions, ADDS-to-ADDS communication, tools for automatic CDB creation, and support for additional DBMSs such as DB2.	Current prototype interfaces with DBMSs on IBM and VAX. Future extensions include graphic query interface, distributed application design tools and performance monitoring. Testing on Transportation Information System.

	MERMAID	MRDSM	ADDS	NDMS
REMARKS	Claims to offer one of the most complete query optimization algorithm that has been implemented and tested. Support replicated and fragmented relations.	Operates in a specialized domain. Not a 'true' heterogeneous system . Heterogenity is dealt at semantic level by providing uniform access to all the databases implemented with same DBMS.	Has been designed to work in the specialized environment of databases relating to oil wells. Current version is being enhanced for corporate wide deployment	
CONTACT PLACE (Person)	UNISYS, 2525 Colorado Ave., Santa Monica, California 90406 (M. Templeton)	INRIA BP 106, 78153 Le Chesnay Cedex, FRANCE (W. Litwin)	Amoco Production Company, Research, P.O. Box 3385 Tulsa Oklahoma 74102 (Glenn R. Thompson)	CRAI Via Bernini S. Rende (CS), ITALY (W. Staniszkis)
REFERENCES	[YU85, TEM86a, TEM86b, MAC85]	[WON84, LIT85, LIT86]	[BRE84, BRE86]	[STA84]

Distributed Heterogeneous Systems.

V. RECOMMENDATIONS AND CONCLUSIONS

The intent of a Distributed, Heterogeneous Database Management System is to provide a logically integrated user interface to physically non-integrated, distributed, hetrogeneous databases. This process of integration encompasses retrieval of information as well as transaction management for multisite updates.

In order to present a single logical database to the user, a global schema is created. Operations on the global schema are translated into corresponding operations on local DBMSs. Creation of a global schema is difficult even when the number of participating databases is small because of several problems [LIT86]. First, The architectures of the local databases vary a great deal from each other [MAN83]. Second, semantic conflicts usually exist between local DBMSs. If the local databases disagree about a value, there may not be a single integrated value satisfactory to all users. Finally, a single global schema may not be possible when the number of local databases is large.

In the area of transaction management, a distributed, heterogeneous database management system involves

incorporation of a concurrency control mechanism and a recovery mechanism, both of which do not interfere with existing mechanisms for local databases. Since one of the objectives of a Distributed, Hetrogeneous DBMS is to provide complete autonomy to local DBMS sites, any change to existing mechanisms for concurrency control and recovery at local DBMSs must be ruled out. A complete solution to the maintenance of global consistency while permitting global-updates, is an area that requires sustained effort [GLI84]. Further, adaptive control techniques must be employed to deal with failures, to support time-critical transactions, and to provide support for replication of information.

In our opinion, specific areas requiring further research work are as follows :

1. Automatic tools for mapping, to cater to various data models, languages, query structures, and data structures.
2. Semantic mapping.
3. System coordination, identification of overall system components and their functions.
4. Query processing, and query optimization.
5. Distributed control over system resources.
6. Synchronization and recovery for multisite updates.
7. Fault tolerance/ failure resistance.

Distributed Heterogeneous Systems.

8. Data security.

9. Communication network services.

10. Modularity.

It appears that advances in broad research areas such as Knowledge-based Engineering, Database Techniques, Computer Graphics, and Distributed Operating Systems will continue to influence and catalyze the growth of Distributed Heterogeneous systems. Through sharing of ideas and concepts, the next generation of distributed heterogeneous database systems should become available in the near future.

REFERENCES

[APE83] Apers, P.M.G., A.R. Henver, and S.B. Yao, "Optimization Algorithms for Distributed Queries", IEEE Transactions on Software Engineering, Jan. 83, pp. 57-68.

[APP85] Appleton, D. S., "The Technology of Data Integration", Datamation, November 1, 1985, pp. 106-116.

[BAR86] Barkmeyer, Edward, Mary Mitchell, K. P. Mikkilineni, Stanley Y. W. Su, and H. Lam, "An Architecture for Distributed Data Management in Computer Integrated Manufacturing", NBSIR 86-3312, NBS, January 1986.

[BER81] Bernstein, P.A., N. Goodman, E. Wong, C.L. Reeve, and J.B. Rothnie, "Query Processing in a System for Distributed Databases (SDD-1)", ACM Trans. on Database Systems, Dec. 81, pp. 602-625.

[BRE84] Breitbart, Y. J. and L. R. Tieman, "ADDS - Heterogeneous Distributed Database System", 3rd Int. Seminar on Distributed Data Sharing Systems, Italy, March 1984, Elsevier, North-Holland, 1985., pp. 7-24.

[BRE86] Breitbart, Y. J., P. L. Olson, and G. R. Thompson, "Database Integration in a Distributed Heterogeneous Database System", IEEE Int. Conf. on Data Engineering, Los Angeles,

February, 1986, pp. 301-310.

[BRO84] Brodie, M.L., "On the Development of Data Models", in On Conceptual Modelling : Perspectives from Artificial Intelligence, Databases, and Programming Languages, M.L.Brodie, J.Mylopoulos, and J.W.Schmidt (Eds) Feb. 84.

[BRO86] Brodie M., "Database Management : A Survey", in Knowledge Base Management Systems, M. L. Brodie and J. Mylopoulos (Eds), Springer-Verlag, 1986.

[DAN82] Daniels, D.P., P. Selinger, L. Haas, B. Lindsay, C. Mohan, A. Walker, and P. Wilms, "An Introduction to Distributed Query Compilation in R*", Distributed Databases, H.J. Schneider (Ed.), Sept. 82, pp. 291-309.

[DAY83] Dayal, U., "Processing Queries over Generalization Hierarchies in a Multidatabase System", Proc. of 9th Int. Conf. on Very Large Databases, Italy, 1983.

[DEE81] Deen, S. M., et al, "The Design of a Canonical Database (PRECI)", The Computer Journal, Vol. 24, No. 3, 1981.

[DEE84] Deen, S. M., R. R. Amin, and M. C. Taylor, "Query Decomposition in PRECI*," 3rd Int. Seminar on Distributed Data Sharing Systems, Italy, March 1984, Elsevier, North-Holland, 1985., pp.92-103

[DEE85] Deen, S. M., R. R. Amin, G. O. Ofori-Dwumfuo, and M. C. Taylor, "The Architecture of a Generalised Distributed Database System - PRECI*", Computer Journal, Vol. 28, No. 3,1985, pp. 282-290.

[GLI84] Gligor, V. D. and G. L. Lukenbaugh, "Interconnecting Hetrogeneous Database Management Systems", Computer, January 1984, pp. 33-43.

[GLI86] Gligor, V. D. and R. Popescu-Zeletin, "Transaction Management in Distributed Hetrogeneous Database Management Systems" Information Systems, Vol. 11. No. 4, pp. 287-297, 1986.

[GOL84] Goldhisch, D., T. landers, R. L. Rosenberg and L. Yedwab, "MULTIBASE System Administratior's Guide", Tech. Rep., CCA, Cambridge, MA, November 1984.

[IIS83] Integrated Information Support System (IISS) report, "Integrated Computer-Aided Manufacturing (ICAM)", Materials Laboratory, Air Force Systems Command, Wright-Patterson AFB, February 1983.

[KRI85] Krishnamurthy, P., "A Data Manipulation Language for the Semantic Association Model, SAM* ", Masters Thesis, College of Engineering, University of Florida, Gainsville, FL, 1985.

[LAN82] Landers, T. and R. L. Rosenberg, "An Overview of MULTIBASE", Second Symp. Distributed Database, Berlin, Sept. 1982, North-Holland, New york, 1982, pp. 311-366.

[LIB86] Libes, Don and Ed Barkmeyer, "IMDAS - An Overview", Draft Report, Integrated Systems Group, NBS, Gaithersburg, MD, 1986.

[LIT85] Litwin, W., "An Overview of the Multidatabase System MRDSM,"ACM National Conf., Denver, october 1985.

[LIT86] Litwin, W. and Abdelaziz Abdellatif, "Multidatabase Interoperability,", Computer, December 1986, pp. 10-18.

[MAC85] MacGregor, R., "ARIEL - A Semantic Front-End to Relational DBMSs", in Proc. of VLDB, August 1985.

[MAD87] Madnick, S.E., et al, "Concurrecy Control and Recovery in Distributed, Hetrogeneous Database Management Systems., working paper, 1987.

[MAN83] Manola, F.A., "Model to Model Mappings and Conversion in a Family of Data Model Specifications", Computer Corporation of America report No. CCA-83-14, Dec. 83.

[SHI81] Shipman, D., "The Functional Data Model and the Data Language DAPLEX", ACM Trans. on Database Systems, March 1981.

[SMI81] Smith, J. M. et. al., "Multibase - Integrating Heterogeneous Distributed Database Systems", Proc. of AFIPS, Vol. 50, 1981, pp. 487-499.

[STA84] Staniszkis, W. Kaminski, M. Kowalewski, K. Krajewski, S. Mezyk, and G. Turco, "Architecture of the Network Data Management System," 3rd Int. Seminar on Distributed Data Sharing Systems, Italy, March 1984, Elsevier, North-Holland, 1985., pp. 57-74.

[SU 83] Su, Stanley, "SAM* : A Semantic Association Model for Corporate and Scientific-Statistical Databases", Information Sciences, Vol. 29, 1983, pp. 151-199.

[TAN85] Tanenbaum, A.S., and R. V. Renesse, "Distributed Operating Systems", Computing Surveys, vol. 17, No. 4, Dec. 85.

[TEM86a] Templeton, M., D. Brill, A. Chen, S. Dao, and E. Lund, "MERMAID - Experiences With Network Operation,", IEEE Int. Conf. on Data Engineering, Los Angeles, February 1986, pp. 292-300.

[TEM86b] Templeton, M. et. al., "MERMAID - A Front-end to Distributed Heterogeneous Databases", Report, System Development Corporation, Santa Monica, CA., 1986

[WON84] Wong, K. K. and P. Bazex, "MRDSM: A Relational Multidatabases Management System", 3rd Int. Seminar on Distributed Data Sharing Systems, Italy, March 1984, Elsevier, North-Halland, 1985, pp 77-85.

[YU 83] Yu, C.T. and C.C. Chang, "On the Design of a Query Processing Strategy in a Distributed Database Environment", Proc. of ACM SIGMOD, 1983, pp. 30-39.

[YU 85] Yu, C, C. Chang, M. Templeton, D. Brill, and E. Lund, "Query Processing in a Fragmented Relational Distributed System: MERMAID,", IEEE Trans. on Software Engineering, August 1985.

BIBLIOGRAPHY

[APP86a] Appleton, D. S., "Very Large Projects", Datamation, January 15, 1986, pp. 63-70.

[APP86b] Appleton, D. S., "Information Asset Management", Datamation, February 1, 1986, pp. 72-76.

[APP86c] Appleton, D. S., "Rule-Based Data Resource Management", Datamation, May 1, 1986, pp. 86-99.

[BRE84] Breitbart, Y. J., L. F. Kemp, G. R. Thompson, A. Silberschatz, "Performance Evaluation of a Simulation Model for Data Retrieval in a Heterogeneous Database Environment", Proc. of IEEE, Trends & Applications 1984, pp. 190-197.

[BRE85] Breitbart, Y. and P. Paolini, "The Multibase Session", in Distributed Data Sharing Systems, F. A. Schreiber and W. Litwin (Eds.) Elsevier (North-Holland), 1985, pp. 3-6.

[CAR85] Cardenas, A.F., and Wang, G.R., "Translation of SQL/DS Data Access/update into Entity-relationship Data Access/update", Proc. of the 4th IEEE Intl. Conf. on Entity-Relationship Approach, Oct. 85.

[CAR86] Cardenas, A.F., "Hetrogeneous Distributed Data Base

Management The HD-DBMS", Submitted for Publication, Oct. 86.

[CHI84] Chimia, J. L., "Query Decomposition in a Distributed Database System Using Satellite Communications", 3rd Int. Conf. on Distributed Data Sharing Systems, Italy, March 1984, Elsevier (North-Holland), pp. 105-118.

[CHU86] Chung, L. V., M. D. Yanike, E. J. Johnson and E. J. Byrne, "Design and Implementation of a Relational Database Server In a Hetrogeneous Network Environment", IEEE Int. Conf. on Data Engineering, Los Angels, California, FEB. 5-7, 1986, pp. 685-692.

[ELM84] Elmasri, R., and Navathe, S., "Object Integration in Logical Database Design", Proc. of Intl. Conf. on Data Engineering, Apr. 84, pp. 426-433.

[ELM86] Elmasri, R., Larson, J. and Navathe, S., "Schema Integration Algoriths for Federated Databases and Logical Database Design", Dept. of Computer Sc., University of Houston, Houston, TX 77004.

[GLI85] Gligor, V. D. and R. Popescu-Zeletin, "Concurrency Control Issues in Distributed Hetrogeneous Database Management Systems", 3rd Int. Seminar on Distributed Data Sharing Systems, Italy, March 1984, Elsevier, North-Holland, 1985, pp 43-56.

[HEI85] Heimbigner, D., and McLeod, D., "A Fedrated Architecture ffor Information Management", ACM Trans. on Office Information Systems, Vol. 3, No. 3, July 85, pp. 253-278.

[KEL86] Keller, A.M., "The Role of Semantics in Translating View Updates", Computer, IEEE magzine, Jan. 86.

[KIM81] Kimbleton, S. R. and P. Wang, "Applications and Protocols", Distributed Systems: Architecture and Implementation, Lecture Notes in Computer Science, Paul, Lampson and Siegert, (eds.) Vol. 105, Springer Verlag, New York, 1981, pp. 308-370.

[LEF84] Lefons, E., and A. Silvestri, "The Use of Multidatabase in Decision Support Systems", Proc. of the 3rd Intl. Seminar on Disrtibuted Data Sharing Systems, Italy, March 84, Elsevier, North-Holland, 1985.

[LIT82] Litwin, W. et al. "SIRIUS Systems for Distributed Data Management", in Distributed Data Bases, H. J. Schneider (ed), North-Holland, 1982, pp. 311-366.

[LOG81] Leveson, N. G. and A. I. Wasserman, "Logical Decentralization and Semantic Integrity in a Distributed Information System", 2nd. Intl. Seminar on Distributed Data Sharing Systems, Netherlands, June 1981, Elsevier (North-Holland), 1982, pp. 243-253.

[LYN83] Lyngbaek, P., and McLeod, D., "An Approach to Object Sharing in Distributed Database Systems", Proc. of 9th Intl. Conf. on Very Large Data Bases, Oct. 83.

[NAV84] Navathe, S.B., Shashidhar, T., and Elmasri, R., "Relationship Merging in Schema Integration", Proc. of 10th Intl. Conf. on Very Large Data Bases, Aug. 84, pp. 78-90.

[NAV86] Navathe, S., Elmasri, R., and Larson, J., "Integrating User Views in Database Design", Computer, IEEE Magzine, Jan. 86.

[PRA86] Pratt, S. J., "The Alchemy Model: A Model for Homogeneous and Hetrogeneous distributed Computing System", Operating Systems Review, Vol. 20, No. 2, April 1986, pp. 25-37.

[SPA81] Spaccapietra, S., B. Demo, A. Di Leva, C. Parent, C. Perez De Celis and K. Belfar, "An Approach to Effective Hetrogeneous Databases Cooperation", 2nd Int. Seminar on Distributed Data Sharing Systems, Netherlands, June 1981, Elsevier (North-Holland), 1982, pp. 209-218.

[TOJ85] Tojo, A. and T. Sato, "Interoperable Database System: A New R&D Project and Its Impact on Multimedia Information Processing Technology", IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database

Management, Miami, FL, November, 1985, pp 336-339.

[WIE86a] Wiederhold, G., and XiaoLei Qian, "Modeling Asynchrony in Distributed Databases", invited paper, Computer Science Department, Stanford University, Stanford, CA 94305, USA., 1986.

[WIE86b] Wiederhold, G., "Views, Objects, and Databases", Manuscipt submitted, Computer Science Department, Stanford University, Stanford, CA 94305, Oct. 86.

END

DATE

FILMED

9-88

DTIC